TRABAJO FINAL DE CARRERA

TÍTULO DEL TFC: Sistema aviónico de instrumentación para adquisición de datos y gestión de buses ARINC

TITULACIÓN: Ingeniería Técnica de Aeronavegación

AUTOR: Alfonso Israel Gutiérrez

DIRECTOR: Oscar Casas Piedrafita

FECHA: 20 de febrero de 2007

Título: Sistema aviónico de instrumentación para adquisición de datos y gestión de buses ARINC

Autor: Alfonso Israel Gutiérrez Puente

Director: Oscar Casa Piedrafita

Fecha: 20 de febrero de 2007

Resumen

La necesidad de realizar un mantenimiento óptimo de las aeronaves está muy presente en la organización de todas las compañías aéreas actuales. Por ello, en los últimos años se ha trabajado de forma muy activa en el desarrollo de nuevos sistemas de instrumentación y test automáticos que garanticen la seguridad y reduzca los trabajos y tiempos de ejecución.

En este documento se presentan los programas realizados para controlar y gestionar mediante el uso de un sistema PXI (NI PXI-6221 y CEI-620) todos los sistemas aviónicos presentes en la aeronave. Por simplicidad, estos sistemas se han clasificado en tres grupos: los sistemas analógicos, los sistemas digitales y los buses de comunicación.

En primer lugar se comentará de forma introductoria la importancia del "testeo" de los aviones y de la evolución de los aparatos que se encargan de realizarlos, y se citará cuáles son los objetivos principales del proyecto. Después se explicará cómo programar los módulos referentes a los sistemas analógicos y de las aplicaciones que permiten el control de los sistemas digitales. En este apartado se explicará también una nueva forma de utilizar los módulos digitales para realizar medidas de sensores analógicos. Seguidamente, se detallará como realizar la programación para la gestión de los buses de comunicación. En concreto para el bus ARINC 429. Finalmente se concluirá el documento exponiendo una serie de conclusiones y con las expectativas de futuro que se esperan.

Title: Avionics systems of instrumentation to data acquisition and management the buses ARINC.

Author: Alfonso Israel Gutiérrez Puente

Director: Oscar Casa Piedrafita

Date: February, 20th 2007

Overview

The necessary to make a best of aircraft maintenance it's present in all actual aerie companies. For this reason, the latest years it has worked with more interest in the new development systems of automatics instrumentation and test that are guaranteed the security and the Works and times of execution are decreased.

In this document it shows the programs that have created to control and to procure with PXI systems (NI PXI-6221 and CEI-620) in all avionics systems that appear in the aircraft. For more easiness, we have classified these systems in three groups: the analogics systems, the digital systems and the bus of communications.

In first place, it will comment the magnitude of test in aircraft and the evolution of machines that do this test. In addition, we say the leading objectives in this Project. Next, we explain how we will program the applications of analogics and digitals systems. Besides, in this section we explain a new form to use the digital modules in the measurement of analogics sensors. Alter, it details the accomplishment of program to manage the bus of communications. In particular to bus ARINC 429. Finally, we will conclude this document whit some conclusions and we say the possible future.

Índice

Introducción 1

1. Gestión de los sistemas analógicos 3
   1.1. Canal Analógico de Salida (AO) 3
      1.1.1. *Driver* realizado 3
      1.1.2. Interfaz del usuario 7
   1.2. Canal Analógico de Entrada (AI) 8
      1.2.1. *Driver* realizado 9
      1.2.2. Interfaz usuario 11

2. Gestión de los sistemas digitales 13
   2.1. Canal Digital Entrada (DI) / Canal Digital Salida (DO) 13
      2.1.1 *Driver* realizado 13
      2.1.2 Interfaz del usuario 18
   2.2. Contador 19
      2.2.1 *Driver* realizado 19
      2.2.2 Interfaz del usuario 28

3. Gestión de los buses de comunicación 30
   3.1. Normativa ARINC 429 30
      3.1.1. Datos de información 31
      3.1.2. Datos de protocolo. 36
   3.2. Gestión de los buses ARINC 429 38
      3.2.1. Programa de transmisión de datos 38
      3.2.2. Programa para recibir datos 43
      3.2.3. Resultados obtenidos 46

4. Conclusiones 49

Bibliografía 50

Anexo I 51
   Características 51

Anexo II 55
   1. Funciones adicionales 55
      1.1. Guardar 55
      1.2. Buscador 60
      1.3. Programa para guardar y cargar automáticamente direcciones y estructuras de palabras. 62
      1.4. Conversión datos decimal a BCD 66
      1.5. Conversión datos BCD a decimal 69
      1.6. Conversión datos decimales a BNR 71
      1.7. Conversión datos BNR a decimal 74

Introducción

Los continuos accidentes aéreos que se han producido a lo largo de la historia, han propiciado que las autoridades gubernamentales creen una serie de normativas que permita asegurar la seguridad de los pasajeros.

Entre estas normativas se encuentran las que obligan a las compañías a revisar cada uno de los componentes del avión.

Esta revisión consta de dos tipos de test. El primer tipo permite comprobar la estructura física. Para ello, se realizan test de pruebas mecánicas en toda la estructura del avión. El segundo test es el que verifica si los sistemas aviónicos funcionan correctamente. Este proyecto se basa en este segundo tipo de test.

En los sistemas aviónicos pueden distinguirse tres tipos de sistemas, los sistemas analógicos, los sistemas digitales y los sistemas de comunicación. Para poder obtener cualquier señal perteneciente a alguno de estos tres grupos, las compañías aéreas utilizan sistemas integrados de adquisición, excitación y control de la señal. Como empresa más destacada en este tipo de sistemas se encuentra “National Instruments” (NI). Para la realización de este proyecto, se ha utilizado un sistema de instrumentación basado en una plataforma PXI de la empresa NI.

Las principales ventajas de estos sistemas son cuatro. La primera es que permite tener un número elevado de canales. La segunda ventaja es que puede testearse automáticamente los sistemas, al poseer sistemas de control digital avanzados empotrados. La tercera es que al ser módulos, se ahorra un importante espacio, en comparación de los antiguos sistemas VXI, y PCI. La cuarta, es que dispone de una gran sincronización interna, mejorando los tiempos de control de trigger interno y la precisión del reloj, lo que permite un correcto funcionamiento de las comunicaciones entre el sistema integrado de adquisición y el ordenador. Aunque actualmente ya existe una nueva plataforma de tamaño más reducido, LXI, aun no está completamente introducida en las aplicaciones aviónicas.

Dependiendo del tipo de adquisición o excitación que se necesite realizar, se utilizará un módulo o placa u otro. El sistema disponible para la realización de este trabajo fin de carrera lleva incorporado las tarjetas NI PXI-6221, NI PXI-8310 y CEI-620. La tarjeta NI PXI-6221 permite controlar las señales analógicas y digitales, mientras que la tarjeta CEI-620 sirve para controlas las comunicaciones entre los diferentes dispositivos del avión. La tarjeta NI PXI-8310 es la que realiza la comunicación entre el sistema integrado de adquisición de datos y un sistema digital portátil.

En un principio, este modelo de adquisición podía llevar integrado un ordenador en el cual se realizase directamente las medidas. Sin embargo, esta opción resultaba demasiado cara. Es por ello, que se ha optado controlar la tarjeta NI PXI-6221 y CEI 620 mediante un sistema digital portátil.

El **objetivo** de este proyecto es implementar un sistema que permita controlar mediante un sistema digital portátil las dos tarjetas mencionadas anteriormente para poder **gestionar los sistemas analógicos, los sistemas digitales y los buses de comunicación de un avión.**

1. Gestión de los sistemas analógicos

Para controlar los sistemas de adquisición y excitación analógicos se ha utilizado la tarjeta NI PXI – 6221. Esta tarjeta dispone de dieciséis entradas analógicas y dos salidas analógicas. Cada canal de entrada dispone de una velocidad de muestreo de 250 KS/s, una resolución de 16 bits y un margen de entrada programable de ±10 v, ±5 v, ±1 v y ±0,2 v, mientras que para los canales de salida, la velocidad de subida es de 833 KS/s y la resolución de 16 bits. Para más especificaciones ir al Anexo I.

El objetivo de esta parte del proyecto es la de crear un programa que permita controlar las funcionalidades descritas anteriormente. El programa se dividirá en módulos y funciones. De esta forma, parte del programa puede ser utilizado en otros posibles programas que realicen funcionalidades parecidas.

1.1. Canal Analógico de Salida (AO)

Este módulo pretende generar tensiones alternas y continuas para excitar los circuitos o sistemas a probar. En la generación de señales alternas, se desea controlar la amplitud, la frecuencia y la forma de la onda. En el caso de la tensión continua, además de modificar la amplitud, se pretende que la alimentación continua pueda ser tanto unipolar como diferencial.

Por otra parte, se desea controlar, tanto para la generación de señales alternas como las continuas, el canal por dónde se transmitirá la señal y la velocidad de transmisión.

En el siguiente apartado se especifica con detalle la programación de estas funcionalidades.

1.1.1. *Driver* realizado

Antes de describir la programación, cabe destacar que para flexibilizar estas aplicaciones se ha decidido utilizar las funciones DAQmx de NI. De esta manera, los módulos pueden ser utilizados para controlar funcionalidades de cualquier tarjeta de adquisición de datos de Nacional Instruments que tenga las mismas características o parecidas.

1.1.1.1. Función de señal alterna

El esquema de su programación puede visualizarse en la Fig. 1.1.

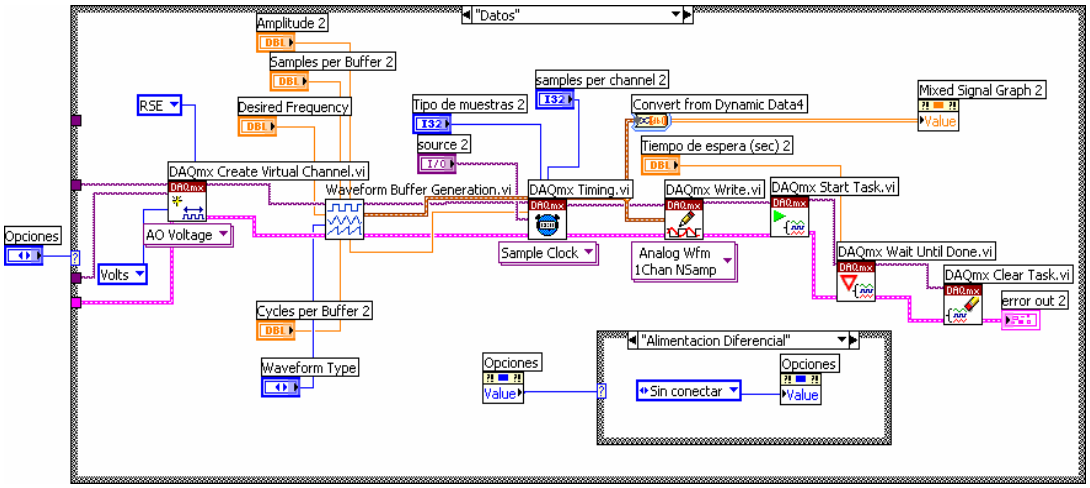

Fig. 1.1 Funcionalidad para generar señales alternas.

A continuación, se describirá la funcionalidad de cada elemento del programa y el lugar donde se sitúa.

Antes que nada se necesitará una función que cree tareas. Esta se llama "DQAmx Create Task". Aunque no es necesaria para la ejecución del programa, permite que este se ejecute de forma más rápida. También se necesita de un controlador que permita la conexión del ordenador con el bus de comunicaciones ("DAQmx Physical Channel"). Este controlador estará conectado a su vez a la función "DAQmx Create Virtual Channel". Dicha función permite crear una tarea a partir de un canal. Las tareas son necesarias para poder conectar las distintas funciones que se quiera controlar de la tarjeta. A parte de la entrada del canal físico, la aplicación "DAQmx Create Virtual Channel" dispone de dos entradas de tensión, una máxima y otra mínima a través de las cuales se puede controlar la resolución de la señal recibida, una entrada de configuración del canal para indicar si es alimentación bipolar o unipolar, otra entrada para indicar las unidades y una salida a parte de la tarea que muestra los errores. En la parte inferior dispone de un menú desplegable que permite escoger el tipo de configuración del canal.

Para generar señales alternas periódicas, se necesita la función "Waveform Buffer Generation", que permitirá crear funciones genéricas y escribirlas por el canal físico. Estas funciones genéricas pueden ser señal cuadrada y señal sinusoidal. A través de esta función se puede controlar la amplitud de la señal, la frecuencia deseada, el número de muestras por buffer y los ciclos por buffer. Si se deseara generar otra forma de señal, se debería sustituir esta función por la nueva diseñada.

La función "DAQmx Writer" se sitúa después de las tres anteriores. Es necesaria para poder enviar la alimentación o los datos por el canal. Consta de

tres entradas y dos salidas. La primera entrada corresponde a la tarea, el segundo a los datos que se desean enviar por el canal y la tercera corresponde a la entrada de errores anteriores. La primera salida es la tarea de salida de la función, y la segunda al error producido dentro de dicha función.

La función descrita anteriormente precisa de un reloj. Para poder manipularlo se conecta un controlador que se encarga del número de muestras, otro que permita controlar número de muestras por segundo, otro para el tipo de reloj que se desee tomar, y por último, un controlador que permita controlar la frecuencia. Este último se puede suprimir perfectamente y conectarle la frecuencia de salida de la función “Waveform Buffer”.

Una vez enviada la función deseada a la función “DAQmx Writer”, este se encarga de enviarlo a la función “DAQmx Star Task”, el cual permite enviarlo físicamente por el bus de comunicaciones. A continuación, se ejecuta la función “DAQmx Wait Until Done”, que define el tiempo a partir del cual el programa envía los datos. Después de la espera, ejecuta el programa de paro de emisión de datos “DAQmx Stop Task”. Por último se inicia el proceso de vaciado del canal de tareas para tenerlo listo para una nueva tarea, “DAQmx Clear Task”. Esta función solo es necesaria cuando se envían datos de forma continua, con lo que se traduce a que las tareas ejecutadas se encontrarán dentro de un bucle “While”. En caso contrario, bastaría con la función “DAQmx Stop Task”.

1.1.1.2. Función de señal continua.

Al igual que en el módulo para controlar la señal alterna, puede verse un esquema de la programación de alimentación unipolar en la Fig. 1.2, mientras que en la Fig. 1.3 se muestra la estructura de la alimentación diferencial.

Para el caso de estas alimentaciones se pueden observar ciertos cambios en la programación. Respecto al apartado anterior, en vez de utilizar la función “Waveform Buffer Generation”, se usa “Expres Simulated Signal”. Esta implementación permite generar señales al igual que la otra con la diferencia de que esta puede generar señales de corriente continua mientras que la otra no. Sin embargo, no se puede controlar las muestras ni los ciclos por buffer. Como se necesita una alimentación de corriente continua, al configurar el “Expres” se debe seleccionar la opción de CD (Direct Current) en la pestaña donde pone tipo de función. Una vez terminado se aprieta OK para que la configuración surta efecto. Después de esto, el “Expres” tendrá una entrada Offset en el que se le podrá introducir la tensión de salida que se quiera, y una salida que hace referencia a los datos simulados de la función. Seguidamente de la aplicación anterior, viene la función “DQAmx Start Taskt”, en el que se introduce directamente después, la función “DAQmx Stop Task”. También se necesitará un conversor de datos dinámicos a datos DBL, para transformar los datos extraídos de .la función “Expres” a valores aptos para situarlos dentro de la gráfica.

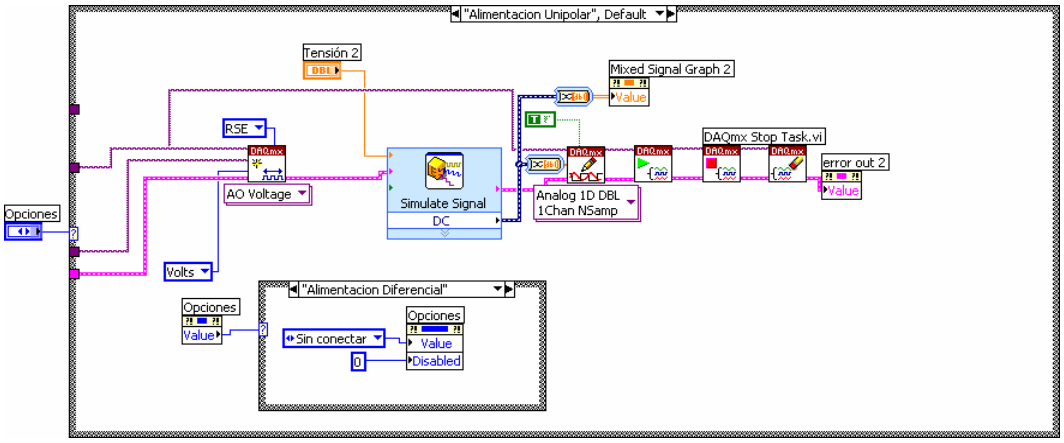

Fig. 1.2 Funcionalidad para generar señales continuas unipolares.

Para el caso de la alimentación diferencial la estructura del programa es la misma, pero con unas pequeñas diferencias. Entonces, la estrategia a seguir para poder enviar información por dos o más canales consiste en indexar los canales deseados e introducirlos en la aplicación "DAQmx Create Virtual Channel". De esta forma se consigue ejecutar más de un canal al mismo tiempo. Sin embargo, esta implementación no permite introducirle arrays. Para ello debe colocarse dentro de un "For Loop". A continuación se enlaza los canales de los puertos a la función. Una vez hecho esto aparecerá un nudo en forma de cuadrado en el extremo del "For Loop". A continuación se pincha con el botón derecho para activar la opción de dar los elementos por índices ("Enable Indexing") y poder enviar los canales de uno en uno. A su vez, el "For Loop" se ejecutará tantas veces como índices se le pasen, siempre y cuando no se le indique el número de veces a ejecutar. Dentro del "For Loop", también debe implementarse con dos "Expres Simulated Signal", Los cuales se les pasará la tensión de cada canal teniendo en cuenta que son el mismo número pero de signo contrario. Las señales de salida de estas dos funciones se indexan a la salida del "For Loop" y se unen al mismo tiempo que se convierten en arrays de tipo DBL. A partir de aquí, la estructura es igual que la anterior.

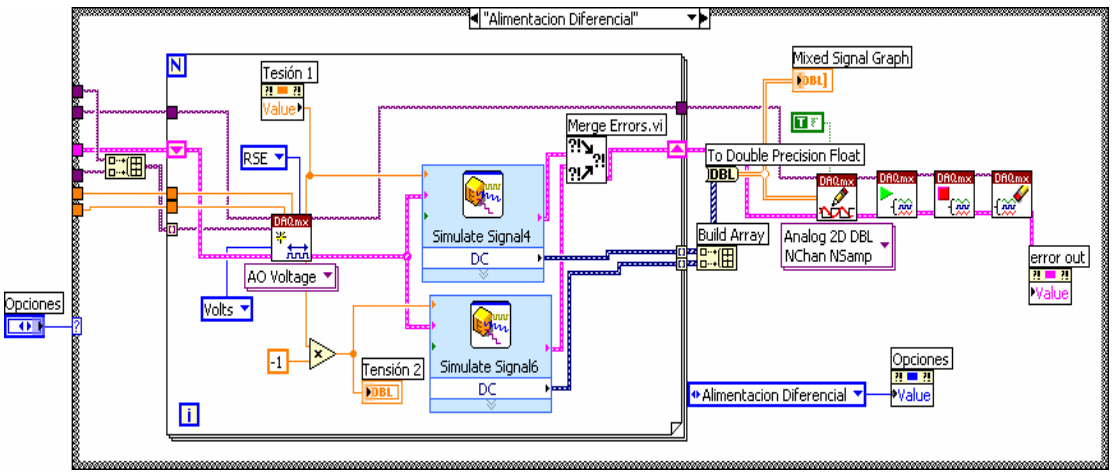

Fig. 1.3 Funcionalidad para generar señales continuas diferenciales.

En un principio, esta implementación sería equivalente a introducir dos o más veces, según el número de canales a leer, las mismas funciones que las que se han explicado anteriormente, sin embargo, con el método anterior, por una parte queda más reducido el programa y por otra parte la ejecución es más rápida.

Al finalizar todas las tareas, se envían los errores producidos y se almacenan en un cluster para mostrarlo por pantalla. Este cluster se compone de un indicador, que muestra el estado y un indicador de texto para mostrar cuál es el problema.

1.1.2. Interfaz del usuario

Para que el usuario pueda controlar fácilmente todas las funcionalidades de las que dispone este módulo, se ha creado una interfaz que permite identificar rápidamente la situación y funcionalidad de cada botón. En la Fig. 1.4 se visualiza la presentación del programa de cara al usuario. A continuación se procederá a describir cada uno de los elementos que se compone la interfaz

En el diseño principal de la interfaz se ha optado por utilizar el controlador “System Tab Control”, que se basa en pestañas, para mostrar el canal cero y uno de la señal analógica.

Se ha elegido esta opción porque permite ver de forma rápida todos los elementos a controlar de cada canal sin necesidad de abrir dos ventanas. Si se hubiera decidido mostrarlo en forma de ventanas se tendría el inconveniente de que no todos los controladores de ambos canales se mostrasen al mismo tiempo. En cambio, si en vez de mostrarlo por separado, se hubiera optado por mostrar los dos canales por la misma pantalla, esta no quedaría tan comprimida con la opción “System Tab Control”. De esta forma se pueden colocar controladores adicionales en una misma ventana. Por esta razón esta opción no sólo se ha utilizo en este programa sino que se ha aplicado para el siguiente programa. Cabe destacar que esta opción puede resultar bastante incómoda si el número de pestañas a implementar es grande.

Para la elección de opciones a realizar en el programa se ha implementado con el controlador “System Radio Buttons” porque permite desactivar una opción cuando se activa otra. Además, también permite añadir más botones que permita tener más de dos opciones, lo que supone una gran ventaja respecto a los de solamente dos. Estas opciones corresponden a “Alimentación Unipolar”, para enviar señales unipolares continuas, “Alimentación Diferencial”, que permite generar señales diferenciales continuas, “Datos”, que generan señales alternas y “Sin conectar”.

Otro elemento esencial que dispone la interfaz por cada canal es una gráfica que permita visualizar los datos o tensiones enviados por el bus en cualquier momento y evaluar su estado. Para la implementación de las gráficas se ha tenido en cuenta que también se leerá más de una gráfica, lo que significa que se introducirán arrays de dos o más dimensiones. Por esta razón se ha escogido la gráfica “Mixed Signal Graph”.

Para ejecutar la opción de alimentación diferencial se ha de procurar que los dos canales se encuentren encendidos a través de los botones grandes que se encuentran situados arriba a la derecha.

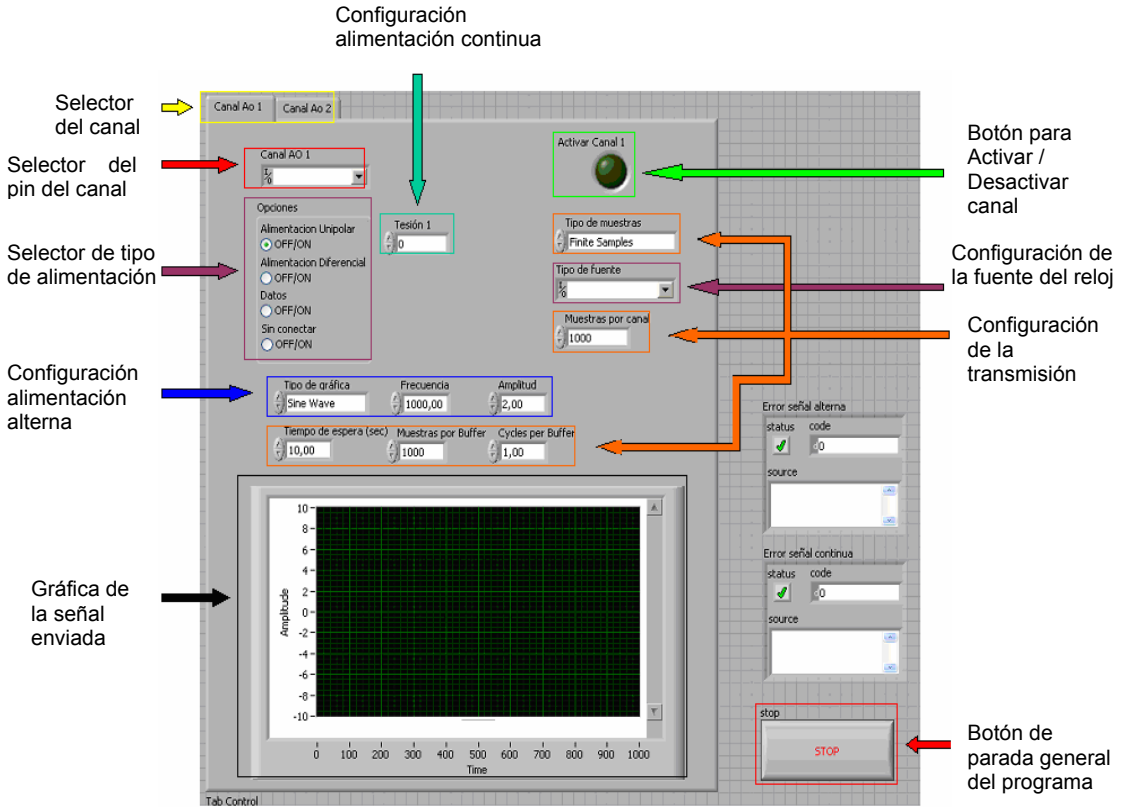

Fig. 1.4 Interfaz de control de los canales de salida analógica.

1.2. Canal Analógico de Entrada (AI)

El fin de este módulo es el de controlar los diferentes canales por donde se adquiera la señal, la velocidad de muestreo y la resolución de la señal que se recibirá. A su vez, también ha de ser capaz de mostrar por pantalla la señal recibida, tanto en formato numérico como en una gráfica. Finalmente, también debe de disponer la opción de guardar los datos numéricos en un fichero, ya sea en formato Word o en formato de texto.

Aunque esta tarjeta sólo es capaz de soportar una alimentación unipolar, se pretende al igual que en el módulo anterior disponer de la opción de recibir la señal de forma unipolar o diferencial. De esta forma, se podrá aprovechar la misma función para otras tarjetas de adquisición de NI que permita el uso de esta funcionalidad.

En el siguiente punto, se explica exhaustivamente los pasos a seguir para programarlo.

1.2.1. *Driver* realizado

Al igual que en el apartado 1.1, se aplican las mismas funciones. De esta forma, se podrá reutilizar de forma transparente, todo o parte del programa en otras tarjetas de adquisición de NI.

Como se ha dicho al principio del apartado 1, este dispositivo cuenta con 16 entradas analógicas. Sí se intentará implementar un módulo con dieciséis canales, por una parte resultaría muy costoso tanto en tiempo como en esfuerzo, y por otra parte, la ejecución del programa se realzaría más despacio. Además, ralentizaría considerablemente el ordenador. Es por eso que se ha optado por implementar un módulo con dos canales.

Una vez descrito las características básicas del programa se pasará a describir la implementación del programa comenzando por la alimentación unipolar.

En la Fig. 1.5 se visualiza la programación de la funcionalidad de recepción unipolar, mientras que en la Fig. 1.6, se ve la programación referente a la recepción diferencial.

2.1.1.1. Adquisición de señales unipolares

Para la alimentación unipolar se debe colocar la función “DQAmx Create Task”. A continuación la función “DAQmx Create Virtual Channel” que permitirá trabajar con un canal físico creando uno virtual. En este caso hay que tener en cuenta que la pestaña que tiene en la parte inferior debe seleccionarse “AI Voltage”. Para seleccionar esta opción se aprieta en la propia pestaña en la cual se desplegará un menú desplegable. En él aparecerá esta opción. Como lo que se va a obtener son muestras, eso significa que los datos se obtendrán en formato array. Por tanto, será necesario implementar el módulo con la aplicación “DQAmx Timing”. Sus opciones aparecerán reflejadas en la interfaz del propio programa. El siguiente elemento a implementar es “DQAmx Start Taskt”, que ejecutará el muestreo y lo enviará a la función de lectura “DQAmx Read”. A su vez, la salida de los datos de esta aplicación se mostrará en una gráfica. Como este indicador también ha de ser capaz de mostrar más de una señal de salida, se implementará con un “Mixed Signal Graph”. Además, se mostrarán en una tabla y se retendrán hasta que se salga del programa o se guarden los datos. Para poder mostrar los datos en tablas se utiliza un conversor de valores array a datos dinámicos, “Convert to Dynamic Data”. Una vez la función de lectura haya adquirido el numero de muestras indicadas, este se detiene y a continuación se ejecuta la aplicación “DAQmx Stop Task”. Después la “DAQmx Clear Task”. Por otra parte esta el control de la resolución de tensión de entrada. Dentro del mismo “Case” donde se encuentra la parte unipolar se implementa otro “Case” con cuatro casos, 0.2 V, 1 V, 5 V, 10 V, que son las tensiones a seleccionar. En cada “Case” se colocará con dos constantes que corresponden a las tensiones. La primera constante, que tendrá el valor exacto de la tensión, se multiplicará por menos uno. De esta forma se obtiene dos números con los mismos valores de signo contrario. Cada uno de ellos se conectará cada una de las entradas correspondientes de la función

"DAQmx Create Virtual Channel". El valor de la segunda constante será algo mayor que la primera ya que se introducirá en los nodos de máximo y mínimo valor de la escala Y del indicador "Mixed Signal Graph". De esta forma la visión de la señal no queda tan ajustada en la gráfica, permitiendo al usuario visualizar los mínimos y máximos.

Por último, en la función "DAQmx Create Virtual Channel" se le añade en la entrada "input terminal configuration" la constante RSE ("Referenced single-ended mode") para indicarle a dicha función que la señal es unipolar.

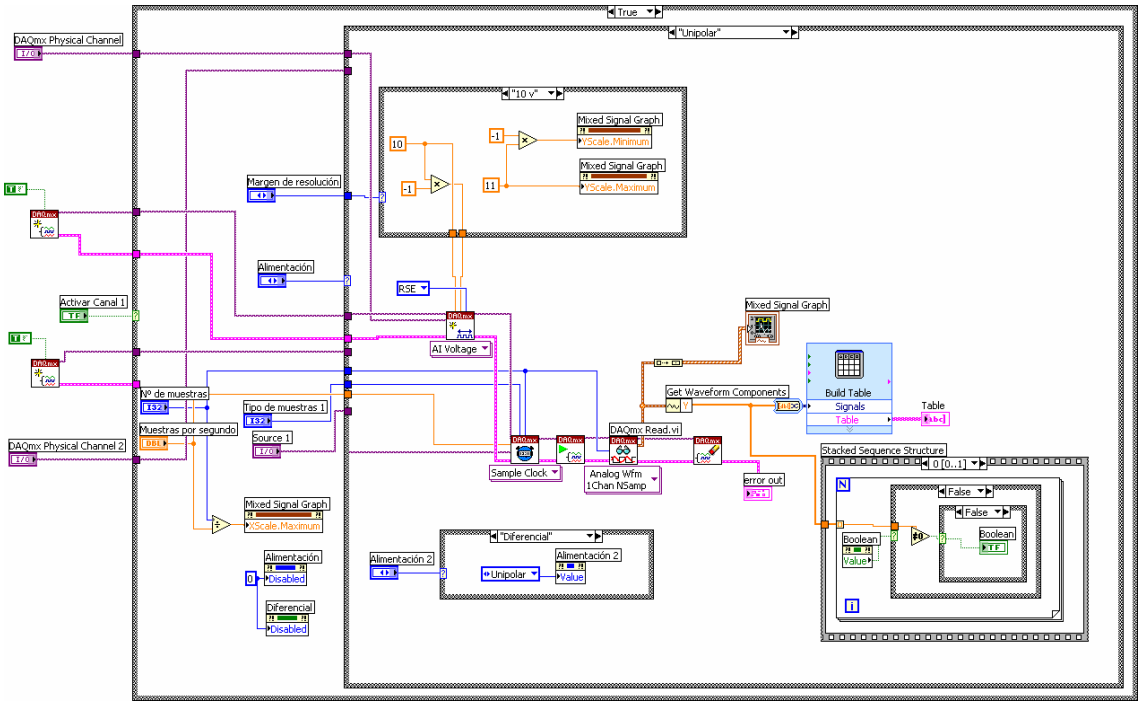

Fig. 1.5 Funcionalidad de adquisición unipolar.

1.2.1.1. Adquisición de señales diferencial

La implementación del caso diferencial es prácticamente la misma que la unipolar solo que en este caso la constante que se le introduce por "input terminal configuration" en la función "DAQmx Create Virtual Channel" es Diferencial. Además, dicha función se encuentra dentro de un "For Loop" que a diferencia del programa para los canales AO, en este caso se utiliza porque el programa no permite ejecutar al mismo tiempo en un mismo espacio, la función "DQAmx Start Taskt" cuando se le entran señales, es decir, que sólo se ejecuta uno u otro. El procedimiento de implementación es el mismo que en el anterior. Por otra parte, los valores dados en el número de muestras, el tipo de muestras, el tipo de fuente usado ("source") y el número de muestras por segundo se transmiten del canal uno al canal dos y viceversa de tal forma que cuando se apriete a la pestaña de un canal u otro, dichos valores se mantengan. También dispone de un nodo de propiedad que corresponde al tipo de alimentación. Cuando en uno de los canales seleccionamos la opción diferencial en el nodo de propiedad, el otro canal también queda activado.

Mientras que si seleccionamos unipolar, el otro toma la posición de unipolar mediante un Case en el que el nodo de propiedad de unipolar queda activado si en el otro canal esta seleccionada la opción unipolar. Por último, contempla la opción de guardar los datos recibidos en un archivo, bien en formato de texto o bien formato Word.

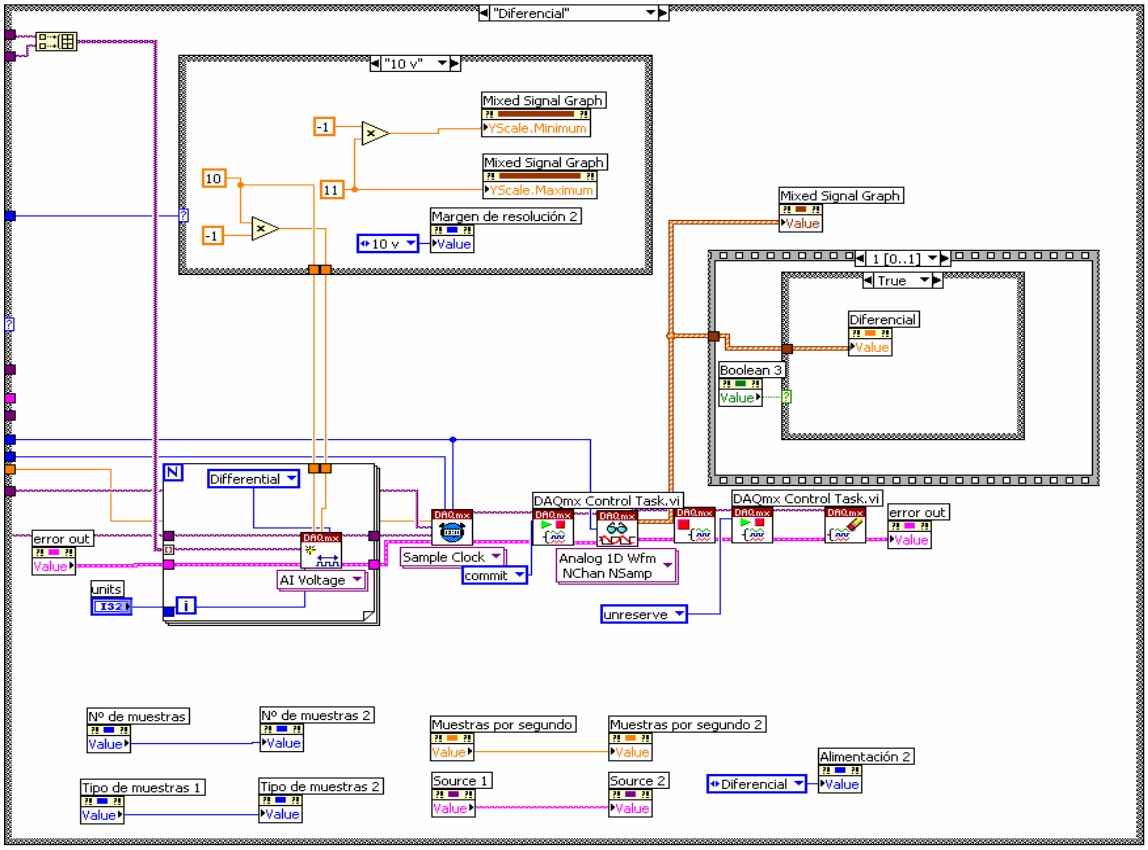

Fig. 1.6 Funcionalidad de adquisición diferencial.

1.2.2. Interfaz usuario

En este apartado se describe la parte del programa que se muestra de cara al usuario. Como en el apartado 1.1.2, se describen las funcionalidades y la situación de cada botón.

En primer lugar puede verse que los canales se seleccionan mediante pestañas, correspondiendo una pestaña para cada canal. En cada pestaña, aparecerá una gráfica donde presentará los datos unipolares o diferenciales. Como en el caso del programa AO, tendremos la posibilidad de encender o apagar cada canal. También se dispondrá como en el programa anterior, de un recuadro con botones que permita seleccionar el tipo de alimentación. Adicionalmente, los valores obtenidos pueden observarse mediante una gráfica o a través de una tabla. Las posibilidad de escoger el canal por el que se quiera recoger la información, el tipo de fuente a utilizar por el reloj, el número de muestras, las muestras por segundo y las muestras por segundo quedarán

plasmadas de la misma manera que en el caso anterior. Por último se dispondrá debajo del botón de puesta en marcha del canal, la opción del margen de resolución.

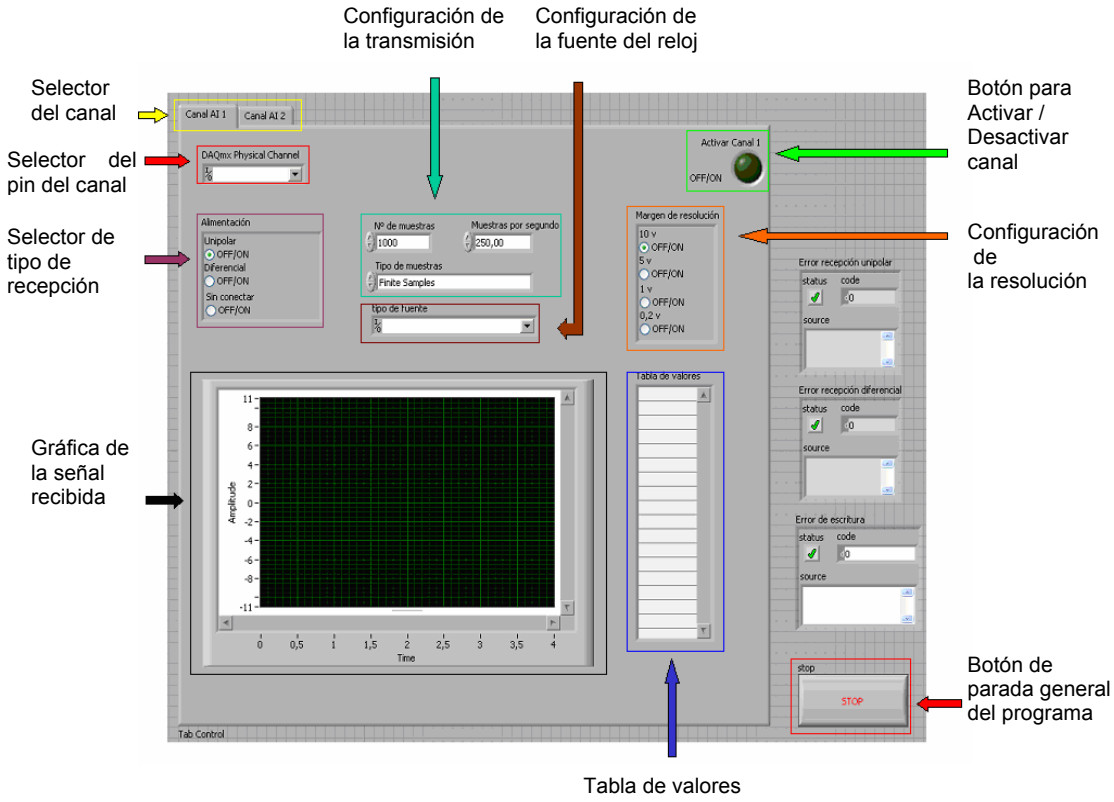

Fig. 1.7 Interfaz de control de los canales de entrada analógica.

2. Gestión de los sistemas digitales

El control de los sistemas de adquisición y excitación digital se realiza a través de la misma tarjeta que en el primer capítulo, la tarjeta NI PXI – 6221. Respecto a la parte digital, esta tarjeta dispone de 24 canales que pueden ser configurados tanto para la transmisión de señales como la adquisición, y una velocidad tanto para la subida como la bajada de 1 Mhz. Además, seis de ellos tienen la posibilidad de configurarse también como contadores. Cabe destacar que cuando se configura los canales para transmitir o adquirir, los canales se denominan líneas. En cada línea envía o recibe un bit. Los canales digitales se componen de ocho líneas. Por cada canal se transmiten o adquieren un byte.

El fin de este capítulo es programar varios módulos que permitan enviar las señales por bits y por bytes de forma manual o a través de un fichero de texto, adquirir señales por bits y por bytes y contar el tiempo de una señal en función del tipo de medida. Además, aprovechando las otras dos interfaces creadas, se ha utilizado de forma innovadora el contador de la tarjeta como elemento de conversión de sensores a resistores.

Los datos adquiridos se guardarán en un fichero de texto, tanto de las señales digitales de entrada, como de los tiempos obtenidos de cualquier señal, como el tiempo de una resistencia en un circuito RC.

2.1. Canal Digital Entrada (DI) / Canal Digital Salida (DO)

La funcionalidad que intenta realizar el primer módulo es la de adquirir y excitar, los sistemas o circuitos conectados. En la adquisición, se pretende recibir bits por las líneas y los bytes por los canales. Los datos obtenidos se guardarán en formato de texto. En la excitación se pretende por una parte, generar y enviar manualmente bits y bytes a través del panel de usuario, y por otra parte, enviar bytes a través de archivos guardados.

2.1.1 *Driver* realizado

De forma análoga al primer capítulo, se ha creado a partir de las funciones DQAmx tanto el módulo del punto 2.1, como el módulo del apartado 2.2. Por tanto, también podrá ser utilizado en otras tarjetas de NI de adquisición de datos con puertas digitales de entrada y de salida.

2.1.1.1 Función de adquisición de señales digitales

La programación de los canales digitales, al igual que los canales analógicos de entrada y salida se implementará de forma similar a los programas anteriores con la diferencia de que en este caso se debe contemplar que el los canales digitales pueden tanto recibir como enviar señales. Primero se comenzará con la opción de recibir los datos, y después se continuará con la opción de enviar datos.

Para recibir los datos, como en los dos programas anteriores, necesitaremos colocar la función “DQAmx Create Task” en la que será precedida por la aplicación “DAQmx Create Virtual Channel”. En la pestaña inferior que dispone, se seleccionará la opción “Digital Input”. Al realizarlo aparecerán cinco entradas y dos salidas. Las que más interesan ya que serán la que se implementen son: entrada de errores, entrada de tareas, entrada del canal físico y agrupación de las líneas (“line grouping”). Este último sirve para especificar si en un canal se envía un bit de información o un byte. Para este caso se seleccionará la opción de un canal para todas las líneas (“one channel for all lines”). Las salidas corresponden a los errores producidos y a la tarea de salida. A continuación se introduce la función “DQAmx Start Taskt” que permitirá ejecutar el recibo de bits. A diferencia que los dos programas anteriores, en este caso no se necesitará la aplicación “DQAmx Timing” porque los bytes serán enviados uno a uno mediante un “While”.

A continuación se muestra un ejemplo de cómo se debería configurar el canal para enviar un byte por canal y de cómo enviar un bit por línea

Ejemplo byte: Dev1/port0/line0:7
Ejemplo bit: Dev1/port0/line0

Como se ha comentado hace un momento, después de la función “DQAmx Start Taskt”, se introducirá un bucle en el que en su interior se hallará la función de lectura DQAmx Read, en el cual se seleccionará la opción “Digital – Single sample – 1D Boolean (N lines)” mediante la pestaña inferior. Este modo permitirá leer los bits de las ocho líneas en forma de variables de booleanas. También habrá un array de ocho booleanas que recibirá los datos que salgan de la función anterior para poderlos mostrar por pantalla en tiempo real, una variable local que corresponderá al Stop general del programa, una función que permita aumentar o disminuir el tiempo de espera entre cada iteración del bucle (“Wait Until Next ms Múltiple”) y un nodo de propiedad que corresponderá al controlador que detenga la lectura sin necesidad de hacerlo con el botón general. La función “Wait Until Next ms Múltiple” en un principio no es fundamental. Sólo se utiliza para que la muestra de los bits por pantalla se realice de forma más plausible. De esta manera el usuario puede realizar el seguimiento del envío.

Por otra parte, el bucle se detendrá cuando la recepción haya terminado, cuando se produzca algún error de lectura, cuando se quiera cerrar el programa directamente mediante el Stop, o si por el contrario no se necesitará obtener más datos o si se quiere pasar directamente de leer datos a recibirlos.

Todos los bytes de salida que se obtengan de la función “DQAmx Read” quedarán acumulados en un índice cuando salgan del bucle. Después del bucle, los datos acumulados se enviarán a un array de dos dimensiones que guardará los datos hasta que finalice el programa o hasta que se guarden. Para eliminar el canal virtual creado, primero se detendrá el envío con la función “DAQmx Stop Task” y a continuación se introducirá la aplicación “DAQmx Clear

Task" en el cual se mostrará a parir de un indicador cluster si se ha producido algún error.

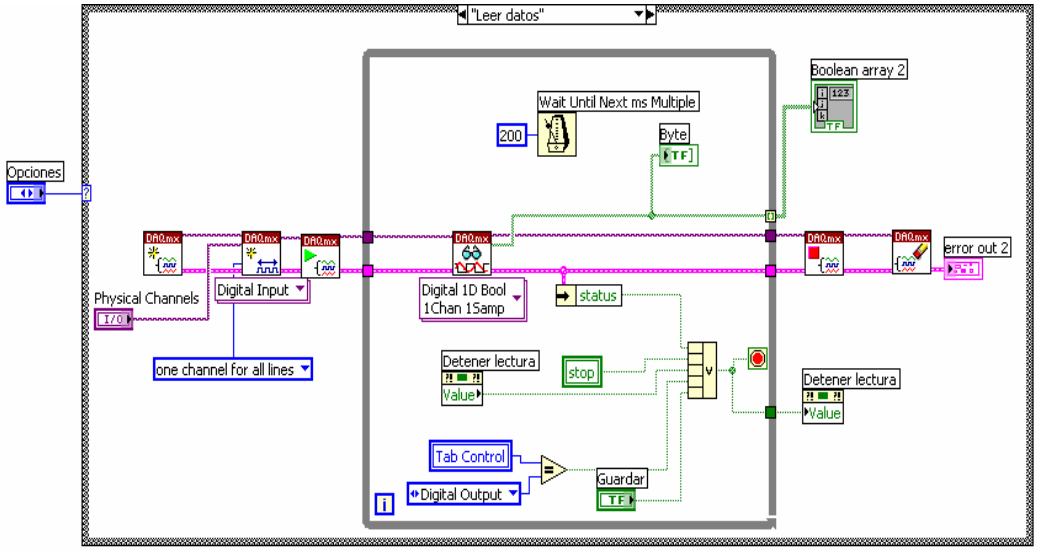

Fig. 2.1 Estructura para adquirir bytes.

Esta implementación se encontrará dentro de la primera secuencia una estructura de tipo secuencia ("Stacked secuence"), la cual irá seguida en la siguiente secuencia de la de la implementación de guardar, es decir, primero se leerán los datos y a continuación se guardarán si es que se desea.

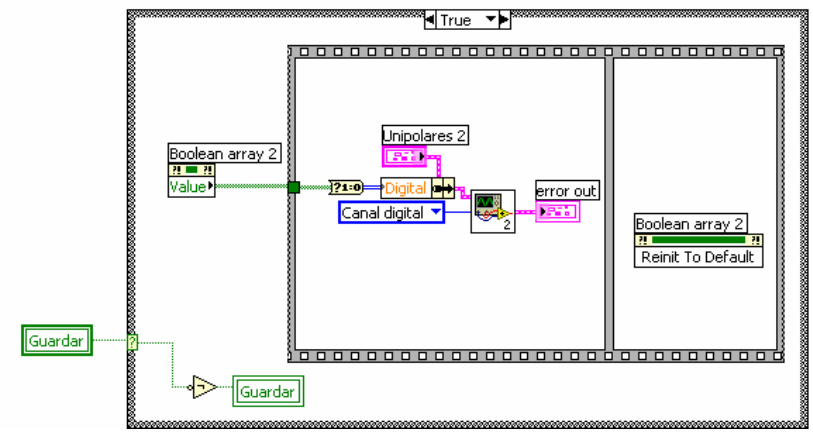

Fig. 2.2 Estructura para guardar bits/bytes.

2.1.1.2 Función de transmisión de señales digitales

La siguiente implementación del programa será el envío de datos. En este caso, cuenta con cuatro opciones; Enviar bit, Enviar byte, Enviar archivo y Detener envío.

La implementaciones de enviar bit y byte son iguales a la de recibir los datos pero con algunas diferencias. En primer lugar, en la función “DAQmx Create Virtual Channel” debemos seleccionar la opción “Digital Output”, y en segundo lugar, la función que se encuentra dentro del bucle es “DAQmx Writer”.

Al igual que en la aplicación “DQAmx Read”, en la pestaña seleccionaremos la opción “Digital – Single sample – 1D Boolean (N lines)” si nuestra intención es enviar bytes.

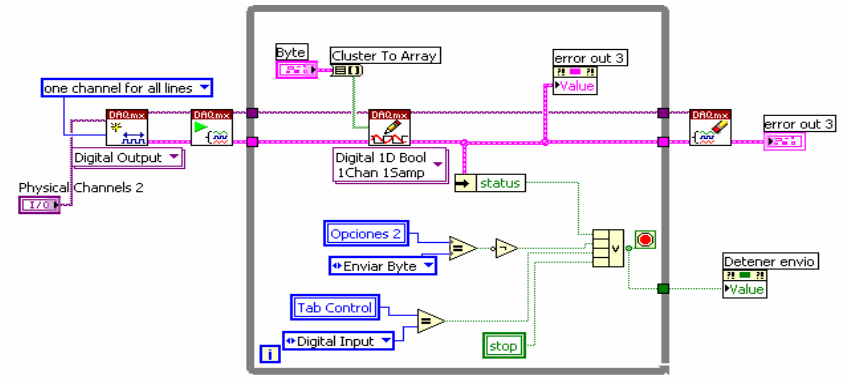

Fig. 2.3 Estructura para transmitir bytes.

Sin embargo, si lo que se desea es enviar un bit, entonces se seleccionará Digital – Single sample – 1D Boolean (1 lines).

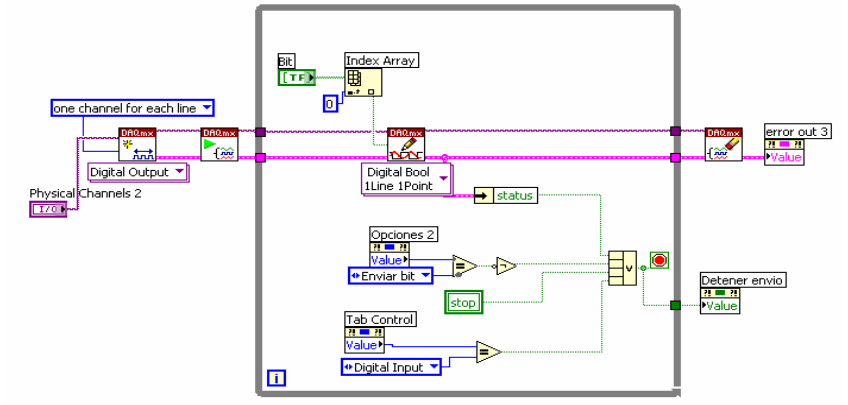

Fig. 2.4 Estructura para transmitir bits.

Otra diferencia que aparece es que para recibir los bytes por pantalla se realiza mediante un cluster que contiene ocho variables de booleanas. La razón por la que se ha decidido cambiar la forma de mostrar los datos, es porque a la hora de representarlo por pantalla en la opción de envío de archivo resulta más sencillo de programar.

La estructura que sigue la opción envío de archivo resulta bastante diferente a las demás. Primero es necesario abrir el archivo en concreto para adquirir los datos. Para ello introducimos la aplicación “Open_Create_Replace File” que permite abrir el documento deseado. En la entrada “function”, introduciremos la constante “open (read only)”, mientras que en la de “prompt”, será una constante de string “Enter Filename”. A continuación, se coloca la función “Read from Text File” el cual lee el documento y extrae toda la información escrita. Después, se cierra el archivo a través de la aplicación “Close File” y se muestran los errores mediante un cluster de errores.

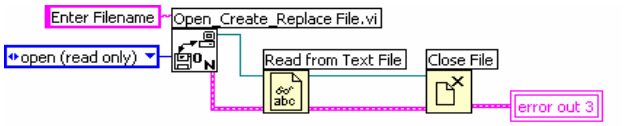

Fig. 2.5 Funciones para guardar datos.

Hay que tener en cuenta que el fichero tiene una cabecera para que el usuario pueda reconocer los valores escritos. Para eliminar esta cabecera y obtener así los datos deseados, se implementan las funciones que aparecen en la Fig. 2.6

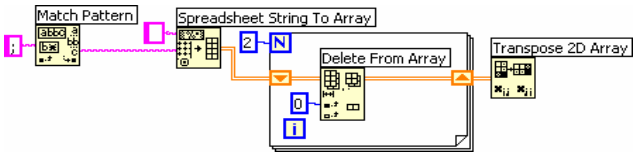

Fig. 2.6 Funciones para extraer y procesar los datos de un archivo.

Los datos obtenidos fuera de de esta función se introducirán dentro de un bucle. Este se ejecutará tantas veces como bytes haya en el archivo o bien hasta que se cambie de opción de salida digital a la de entrada, o se detenga el programa, o bien si se cambia de opción de envío de datos. Dentro del bucle, se obtendrán los datos por columnas, es decir por bytes, a través de la función “array subset”. Los bytes obtenidos se colocarán dentro de un “For Loop” y se seleccionará en el punto de entrada la indexación de los datos para sustraer los bytes no deseados de los que se van a mostrar por pantalla. Finalmente el byte introducido se va mostrando bit a bit por las ocho booleanas que se encuentran colocadas cada una en una posición del Case.

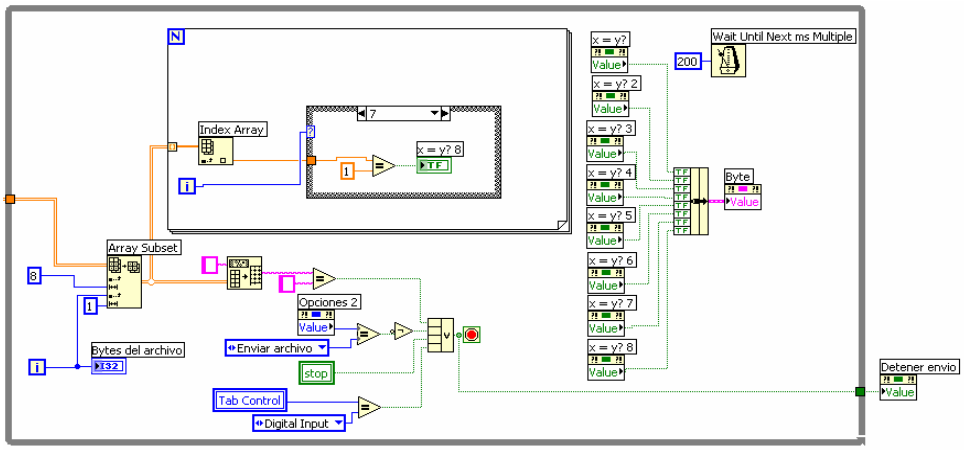

Fig. 2.7 Estructura de transmisión de bits/bytes desde archivo.

2.1.2 Interfaz del usuario

Las funcionalidades descritas en el punto 2.1, se muestra en la Fig. 2.8 para la adquisición, y en la Fig. 2.9 para el envío. En los siguientes párrafos se indica de forma más detallada los controladores usados y cuales son sus funciones.

La presentación del programa de cara al usuario es bastante más sencilla que los anteriores programas. En primer lugar nos encontramos con un "Tab control" que permite el cambio de entradas a salidas digitales de forma rápida.

Para el caso de entrada digital aparece el controlador "System radio buttons" con las opciones de "lectura de datos" y "detener lectura", el controlador del canal físico donde marcaremos las líneas por donde transmitir, un botón que permita guardar los datos cuando se quiera y un array de booleanas que permitirá al usuario seguir la lectura de la recepción.

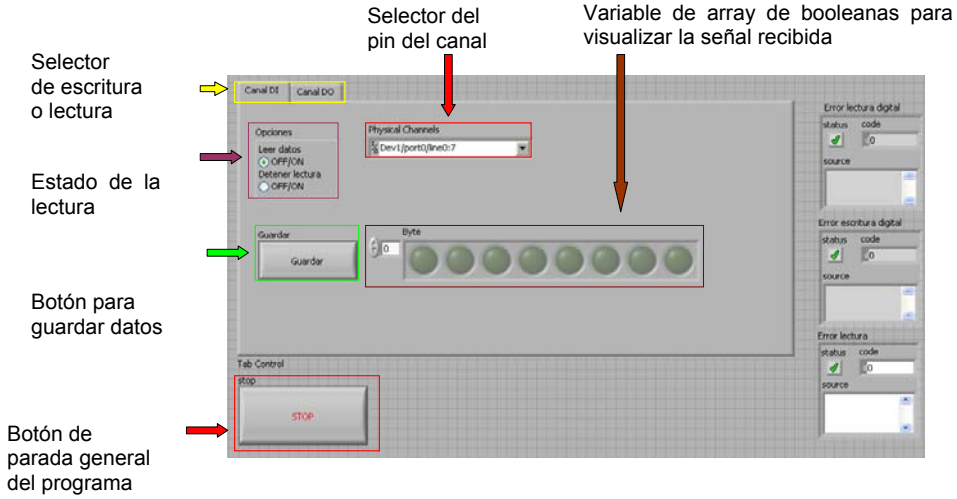

Fig. 2.8 Interfaz de la adquisición de señales digitales.

Si la opción que se escoge es salida digital, entonces el controlador de "System radio buttons" constará de cuatro opciones, "envío de bit", "envío de byte", "envío de archivo" y "detener envío". Al igual que en el caso anterior, también contamos con la opción de escoger el canal de transmisión. En este caso, en vez de aparecer ocho variables de booleanas, aparecen nueve. Uno de ellos corresponde al envío de un bit, mientras que los ocho restantes muestran la transmisión byte a byte. Adicionalmente, también cuenta con un indicador que permite al usuario saber cuantos bytes se están enviando.

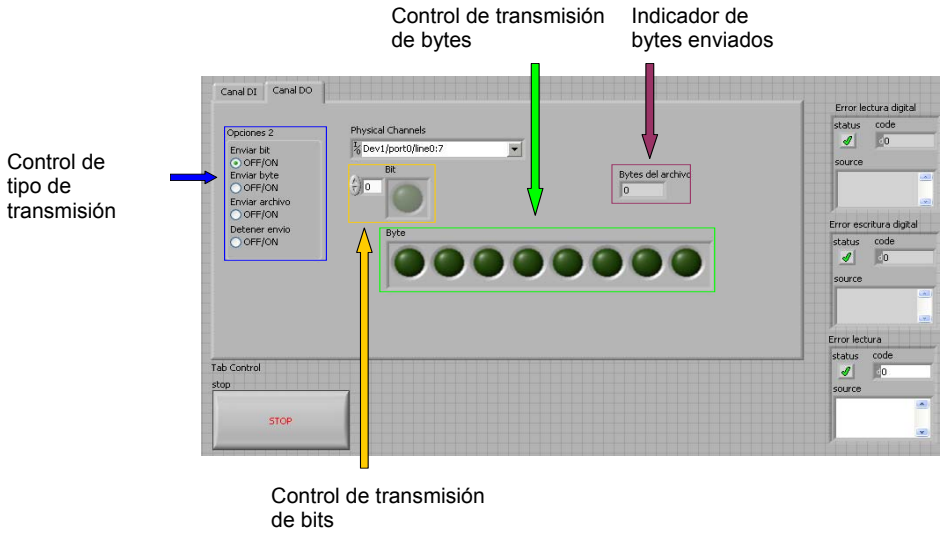

Fig. 2.9 Interfaz de la transmisión de señales digitales.

2.2. Contador

En este modulo se pretende adquirir el tiempo de duración de los pulsos, del periodo, del semiperiodo, la cuenta de eventos, la diferencia temporal entre dos eventos, cada uno de ellos perteneciente a una señal independiente, y la diferencia entre dos posiciones.

Por otra parte se intentará implementar la funcionalidad de medir el tiempo de carga de un condensador.

La última funcionalidad que deberá realizar es la de guardar los datos adquiridos en formato de texto.

2.2.1 *Driver* realizado

2.2.1.1 Funcionalidad contador de retardo de una señal

La última funcionalidad que falta por programar de la tarjeta NI PXI - 6221 es el contador interno con el que cuenta la tarjeta. Si se mira las

especificaciones del contador se ve que puede medir el tiempo del periodo y semiperiodo de la señal introducida, la duración del pulso, los eventos, es decir, las subidas o bajadas de la señal, bien con una señal o bien con dos para el caso de medida de dos eventos y la posición de la señal tal y como se muestra en la Fig. 2.15, debajo del esquema de programación

Además de estas funcionalidades, se ha añadido otra aplicación que ayudará a calcular las resistencias. Esta funcionalidad se explicará después de haber comentado las anteriores funcionalidades.

Se comenzará por implementar la medida de eventos. Para implementar esta función necesitamos como en las aplicaciones anteriores la función "DQAmx Create Task" y "DAQmx Create Virtual Channel". Para que pueda crear un canal virtual específico para el contador, en la pestaña inferior seleccionamos "Counter Input" y en el siguiente menú "Count Edges". El siguiente elemento es "DQAmx Timing". En su parte inferior seleccionaremos la opción "Sample Clock". A continuación, la aplicación que le sigue es "DAQmx Read", en el cual, en su parte inferior se debe seleccionar "Counter – Multiple Samples – 1D DBL". La salida de los datos se introducirá en la variable unipolar. Esta variable es de tipo cluster. En concreto, se seleccionará dentro del cluster la variable unipolar 1. En este apartado se podrá controlar la cuenta de los eventos cuando la señal suba o cuando baje. También se puede indicar a partir de que número queremos que empiece a contar y la dirección en la que cuenta.

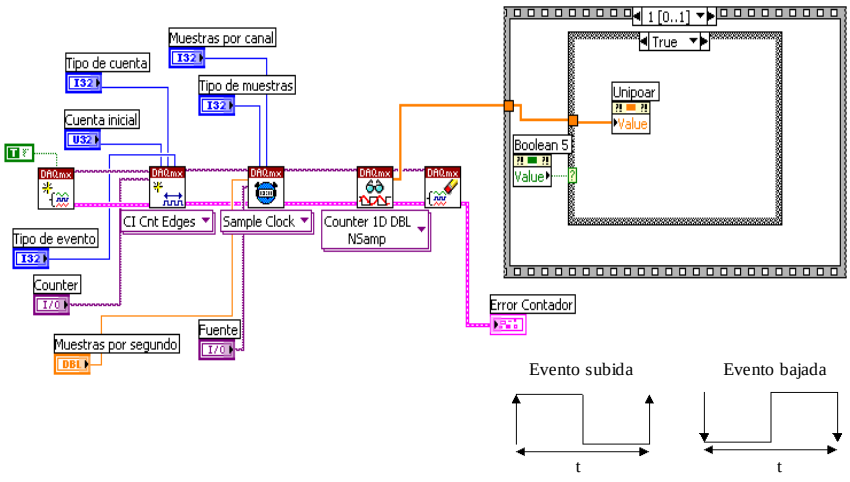

Fig. 2.10 Estructura para medir eventos.

La siguiente función a describir es el contador de pulsos. Su función básica es contar la duración de un pulso. La estrategia a seguir para implementar esta aplicación es la misma que la anterior pero con una serie de variaciones. En primer lugar, en la pestaña inferior de la función "DAQmx Create Virtual Channel" se escogerá "Pulse width". En segundo lugar, se elige la opción "Implicit" en la aplicación "DQAmx Timing". Por otra parte, sólo se podrá indicar que mida cuando la señal se encuentre en el nivel alto, o cuando

este en el nivel bajo. Para el caso de “DQAmx Timing”, únicamente es posible modificar las muestras por canal y el modo en que se muestrea. Los datos que se extraigan se guardarán temporalmente en la variable unipolar 2, que también está vinculado al cluster unipolar.

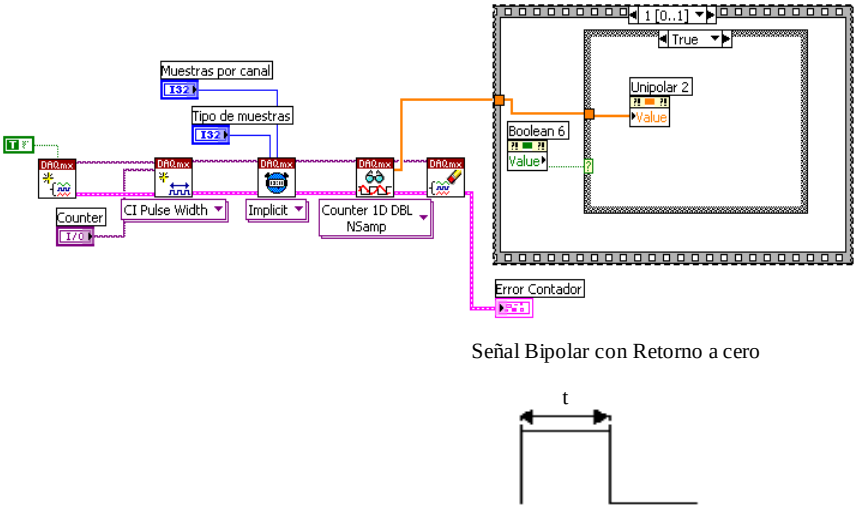

Fig. 2.11 Estructura para medir duración de pulsos.

Para la medida del semiperiodo escogeremos en la función “DAQmx Create Virtual Channel” la opción “Semi Period” que se encuentra en la pestaña inferior. La forma y las funciones restantes se implementan de la misma forma que la aplicación “Pulse width”, con la diferencia que en este caso no podremos controlar los eventos, que los datos se guardarán en la variable unipolar 3 y que en la pestaña de la función mencionada anteriormente se marcará “Semi-period”.

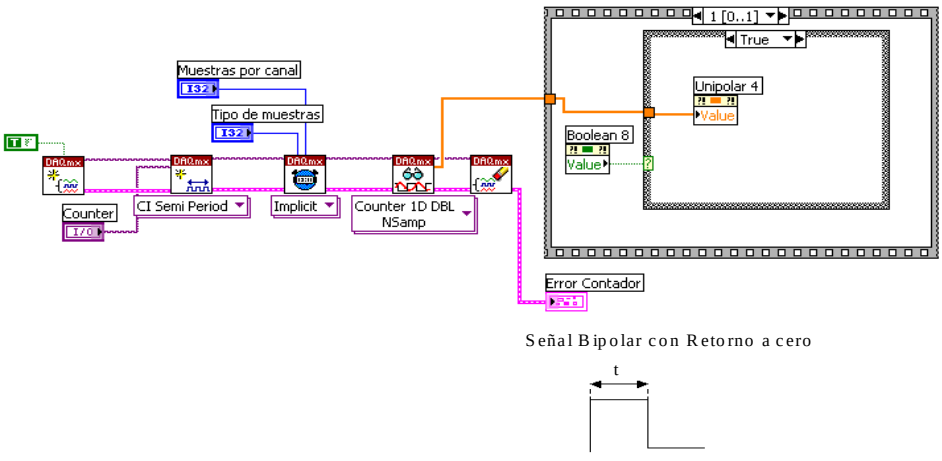

Fig. 2.12 Estructura para medir semiperiodos.

El caso de la medida del periodo es igual que antes, pero ahora se debe elegir en la pestaña inferior de la función “DAQmx Create Virtual Channel” la opción “Period”. La variable donde se guardarán los datos obtenidos de esta aplicación es unipolar 4.

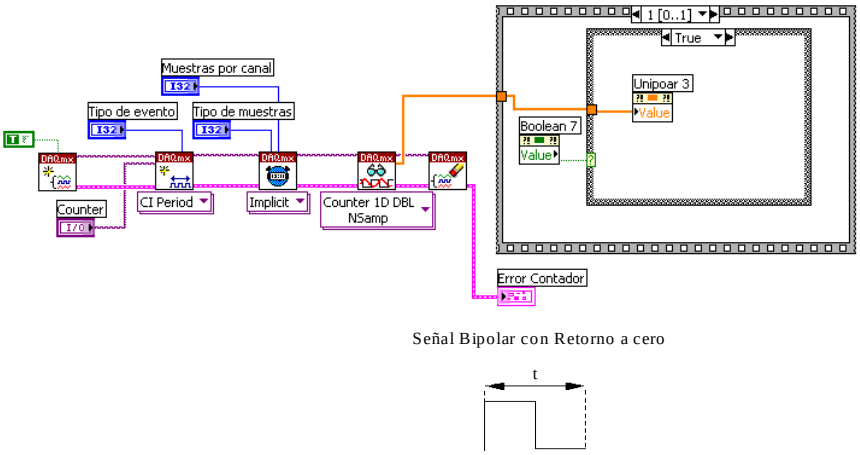

Fig. 2.13 Estructura para medir periodos.

La medición de dos eventos también se implementa de la misma forma que en los dos casos anteriores pero con la diferencia de que en la pestaña de “DAQmx Create Virtual Channel” se escogerá “Two Edge Separation”. Al igual que en la medición de eventos, se puede controlar la medición mediante la selección de los niveles de cada señal, es decir, si deseamos medir una diferencia entre dos niveles altos, en el controlador se deberá seleccionar dicha opción. Los datos obtenidos se guardarán en la variable unipolar 5.

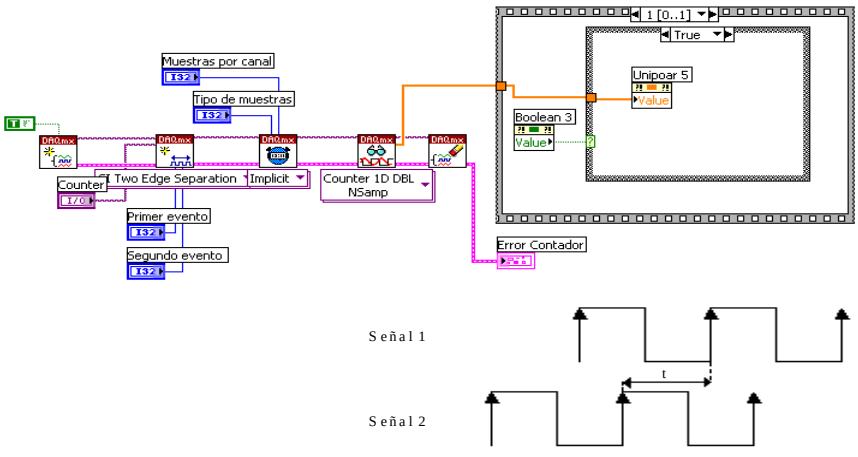

Fig. 2.14 Estructura para medir diferencia de dos eventos.

Para medir el tiempo según la posición de la señal, usaremos la misma estructura que se ha utilizado en la medición de eventos. La única variación es la pestaña inferior de la aplicación “DAQmx Create Virtual Channel” en la que

se escogerá la funcionalidad “Position – Angular Encoder”. En esta parte del programa es posible controlar el valor y la fase del índice Z, el ángulo inicial y los pulsos por revolución. Sus datos se guardarán en la variable unipolar 6.

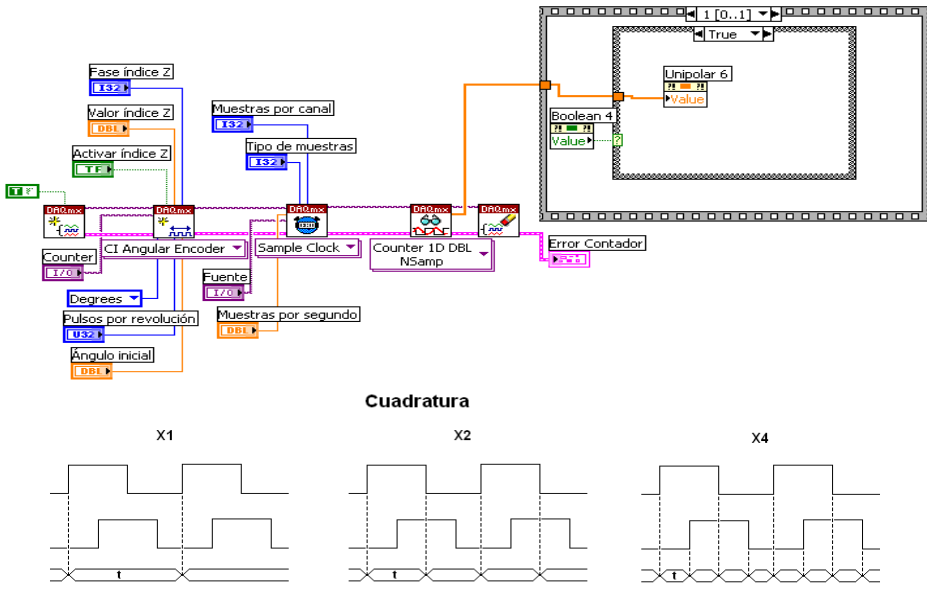

Fig. 2.15 Estructura para medir diferencias de posición.

Cada una de estas funcionalidades descritas se encuentran dentro de un caso (“Case”). Al mismo tiempo, todos estos casos se encuentran dentro de un “While” que repetirá la estructura seleccionada hasta que se detenga el programa.

2.2.1.2 Medida de resistencia mediante interfaz del contador.

Como se ha dicho al principio de este apartado, a parte de programar la funcionalidad básica de la que dispone, se ha intentado crear un programa que permita contar el tiempo de carga y descarga de un condensador en un circuito RC. Si se sabe de antemano el valor del condensador utilizado se podrá obtener el valor de la resistencia que tenemos en nuestro circuito. Para ello se usará la fórmula siguiente:

$$Vi = Vf\left(1 - e^{-\frac{t}{RC}}\right) \qquad \textbf{(2.1)}$$

Para validar la veracidad del sistema, se representaran en una gráfica dichos datos, donde en el eje x aparecerá los valores temporales reales de las resistencias y en el eje y los valores calculados mediante la fórmula anterior sabiendo la resistencia medida.

Hay que tener en cuenta que este tipo de medida únicamente es aplicable para el periodo y el semiperiodo.

La idea para implementar este programa para el caso del semiperiodo es enviar un pulso con valor uno lógico por uno de los canales digitales y a partir de ese momento, comienza a contar con el contador hasta que llegue a otro canal digital. Para el caso del periodo la estrategia es la misma pero con la diferencia de que el contador se detiene cuando vuelve a recibir bit cero.

La señal a medir por el semiperiodo se muestra en la Fig. 2.16, en el apartado (a), mientras que la del periodo que es la misma figura anterior, corresponde al apartado (b).

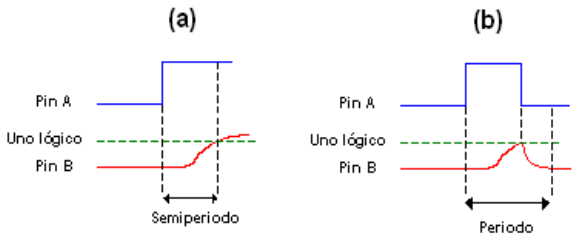

Fig. 2.16 Señal a medir con el periodo y semiperiodo.

El esquema principal de la conexión es el que se visualiza en la Fig. 2.17. Debido a la baja impedancia de entrada que tienen los canales digitales, a la entrada del Pin B se ha colocado el circuito operacional Schmitt Trigger.

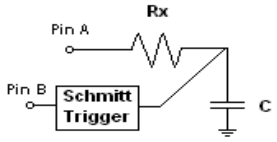

Fig. 2.17 Circuito equivalente de medida

A través del Pin A, se introduce el pulso, mientras que por el Pin B se recibe la señal entre la resistencia y el condensador. A continuación se procede a explica la programación.

Para el caso del semiperiodo, se introducirá un “While” que permitirá realizar el número de muestras que se desee. Como se ha citado anteriormente, dentro del “While” también estará incluido la función “Wait Until Next ms Múltiple”. Además, consta de una estructura de secuencia (“Stacked Sequence Structure”) con cuatro casos. El primer caso corresponde al envío del bit uno lógico, el segundo caso a la medición del semiperiodo y a la llegada del bit, el tercer caso se envía bit cero para descargar el condensador, y el cuarto caso permite comprobar que el condensador se ha descargado satisfactoriamente.

Para el primer caso la estrategia a seguir para implementar dicho caso es la misma que en el canal digital a la hora de enviar un bit. Sus funciones son: “DQAmx Create Task”, “DAQmx Create Virtual Channel” con la opción Digital Output, “DAQmx Read” seleccionando en su pestaña inferior “Digital – Single Channel – Single Sample – Bolean (1line)” y “DAQmx Clear Task”. En este caso no es importante el valor de tiempo de espera puesto que todavía no se ha comenzado a adquirir los datos.

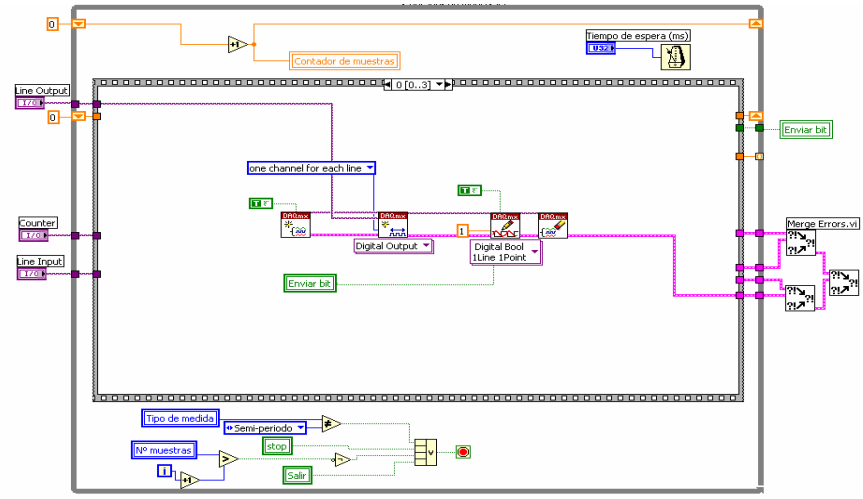

Fig. 2.18 Esquema de la emisión del bit 1 (semiperiodo).

En el segundo caso, se dispondrá de las funciones que permitan adquirir los bits por el canal de entrada, y por otra parte se encuentra las aplicaciones referidas a la medida del tiempo. Las funciones de correspondientes a la lectura de datos para cada uno de los casos se encontrara dentro de un “While” que se detendrá cuando el bit se reciba, cuando se pare el programa o cuando se cambie de opción. El tiempo de espera que se introduce en estas dos funciones ha de ser lo más pequeño posible para evitar una distorsión en la toma de medidas del tiempo.

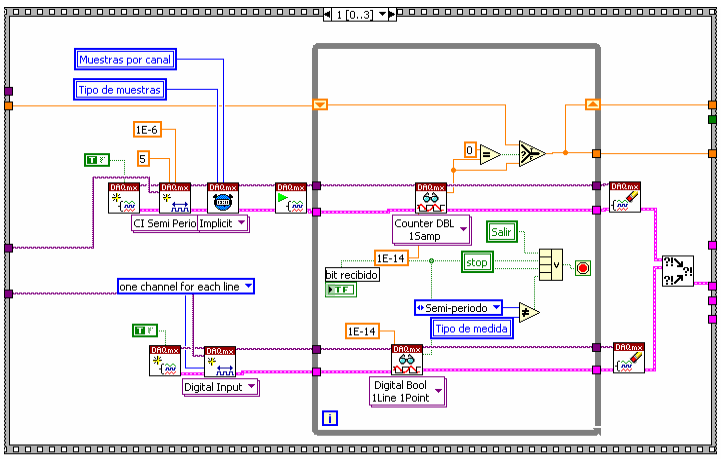

Fig. 2.19 Esquema recepción bit 1 y de la cuenta del retardo temporal (semiperiodo).

El tercer caso se implementa de la misma manera que el primero con la diferencia que en vez de enviar bit uno, transmite bit cero, mientras que el cuarto caso es igual al segundo caso pero sin las funciones del contador.

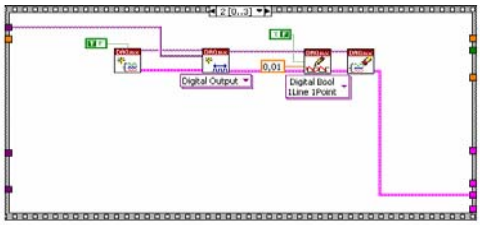

Fig. 2.20 Estructura de transmisión bit 0 (semiperiodo).

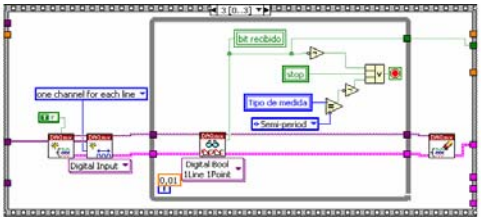

Fig. 2.21 Estructura recepción bit 0 (semiperiodo).

Todas las muestras que se obtengan se guardarán en la variable cluster diferencial, en concreto, en la variable DBL diferencial cuatro.

Para la aplicación de la cuenta de tiempo de la resistencia en el caso del periodo, las funciones a utilizar son similares, sólo que en este caso se mide el retraso de la señal cuando se envía uno lógico y cero lógico. En la Fig. 2.22 se visualiza la estructura entera.

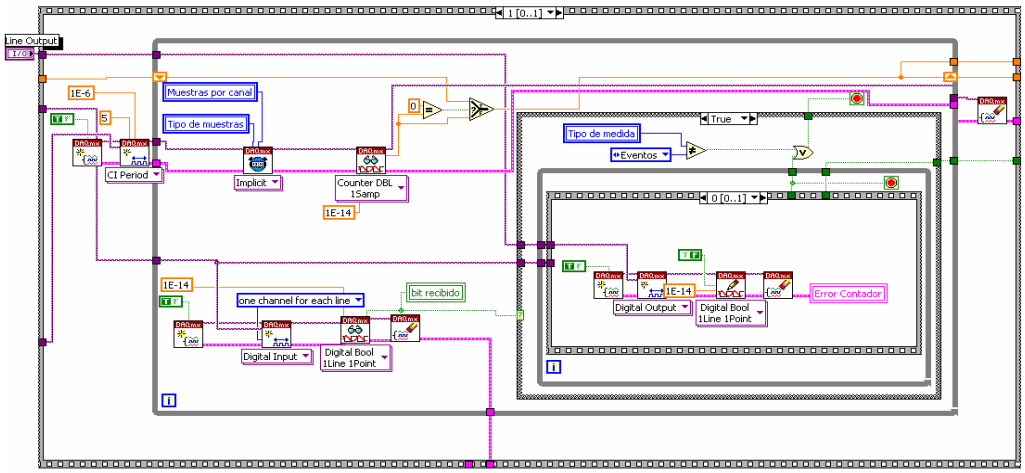

Fig. 2.22 Esquema recepción bit 1, transmisión bit 0 y cuenta retardo temporal (periodo).

La recepción del bit cero se realiza dentro del segundo bucle, después del envío de este por el circuito. Debajo se muestra su estructura.

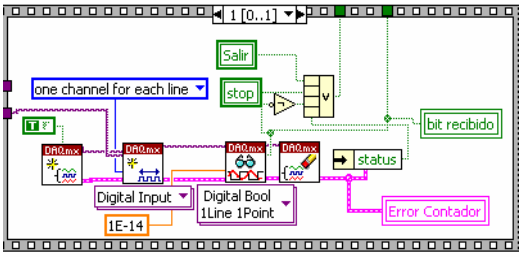

Fig. 2.23 Estructura recepción bit cero (periodo).

Si toda la transmisión y recepción es correcta, se almacenará en la variable diferencial tres.

Los datos guardados en la variable diferencial tres y cuatro, se introducirán en la función “Guardar”, obteniendo un error de ella si no se han realizado de forma satisfactoria su estructura interna.

La gráfica obtenida para cada uno de los casos es prácticamente una recta, es decir, hay una relación lineal entre el tiempo obtenido directamente por el propio contador de la tarjeta y el obtenido mediante la fórmula. Por tanto, puede utilizarse el reloj propio de la tarjeta para medir resistencias en un circuito RC sabiendo la capacidad del condensador. Únicamente debe tenerse en cuenta los errores que puedan producirse por offset, inclinación de la recta y/o posible no linealidad.

Para finalizar con este punto, se mostrará a continuación las gráficas de los tiempos obtenidas a partir de cien muestras realizadas para las siguientes resistencias: 10 kΩ, 27 kΩ, 39 kΩ, 75 kΩ, 91 kΩ, 100 kΩ, 120 kΩ, 150 kΩ, 200 kΩ, 240 kΩ, 300 kΩ.

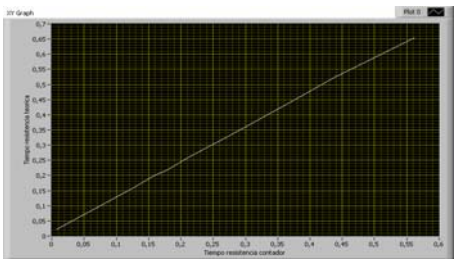

Fig. 2.24 Gráfica tiempo resistencia teórica – tiempo resistencia contador (periodo).

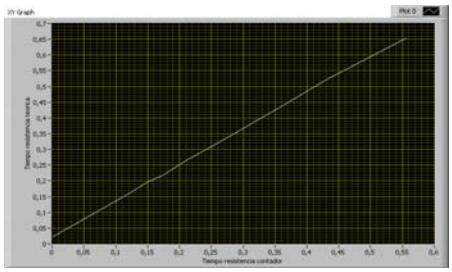

Fig. 2.25 Gráfica tiempo resistencia teórica – tiempo resistencia contador (semiperiodo).

2.2.2 Interfaz del usuario

En la Fig. 2.26 se visualiza todos los controladores que permiten realizar las funciones indicadas en los puntos 2.2.1.1 y 2.2.1.2. Las funciones que controlan cada uno de los botones y controladores se describen a continuación.

En la interfaz se muestra dentro de un cuadro con doble línea los elementos se pueden controlar, mientras que fuera se encuentra el botón de parada general del programa y la muestra de errores tanto para el canal que permite enviar y recibir datos, como para el canal que se encarga de medir los tiempos.

Dentro del cuadro aparece en la parte superior izquierda el controlador que permite seleccionar el canal del contador denominado “Counter”, mientras que en la parte inferior izquierda se visualiza el controlador del canal digital de entrada y de salida, Line Output y “Line Input. Debajo del controlador Counter se haya el controlador que permite seleccionar el tipo de medida a realizar, que como se ha dicho al principio de este apartado son, Eventos, Pulsos, Semiperiodo, Periodo, Separación de dos eventos y Codificación angular. Para ello se ha utilizado el controlador “Systems Radio Buttons” ya que es una forma sencilla de determinar el caso se va a ejecutar dentro del programa y al mismo tiempo fácil de utilizar para el usuario.

Los elementos a controlar en cada una de las medidas se han separado en tres recuadros, que corresponden al “Reloj”, a la “Medida de posición” y a la “Medida del contador”.

Dentro del “Reloj”, se puede controlar los tipos de muestras que se realizan, las muestras por canal, la fuente y las muestras por segundo. Cabe destacar que la fuente y las muestras por segundo sólo se pueden especificar para el caso de Eventos y de Codificación angular.

En la “Medida de posición” solo es aplicable cuando se selecciona Codificación angular. En él se controla por una parte la activación del índice Z, su fase y su valor y por otra parte los pulsos por revolución.

Para el caso de la Medida del contador hay que decir primero que la única medida que no se controla en este recuadro es la Codificación angular porque viene determinado en el cuadro anterior. Dentro de él se encuentra el control del valor máximo y mínimo del tiempo a medir, tipo de evento que sólo

afecta a la medida de pulsos, periodos y eventos, cuenta inicial y tipo de cuenta que únicamente se aplica para el caso de la medida de eventos, y por último, primer evento y segundo evento. Controlan el mismo elemento que tipo de evento, pero en este caso sólo sirve cuando se desea medir la separación de dos eventos.

Los datos obtenidos de la medición se muestran en la columna que se encuentra a la derecha con el nombre de Datos. Los cuatro elementos que se muestran debajo de la columna corresponden a la medida de resistencias. El primer elemento permite controlar el tiempo de espera entre cada muestra para permitir una descarga correcta del condensador. El segundo elemento controla el número de muestras a realizar. El tercer elemento indica el número de muestra que está realizando. El último elemento permite introducir el valor de la resistencia. Este último se verá reflejado en el documento en el cual se haya guardado todos los datos.

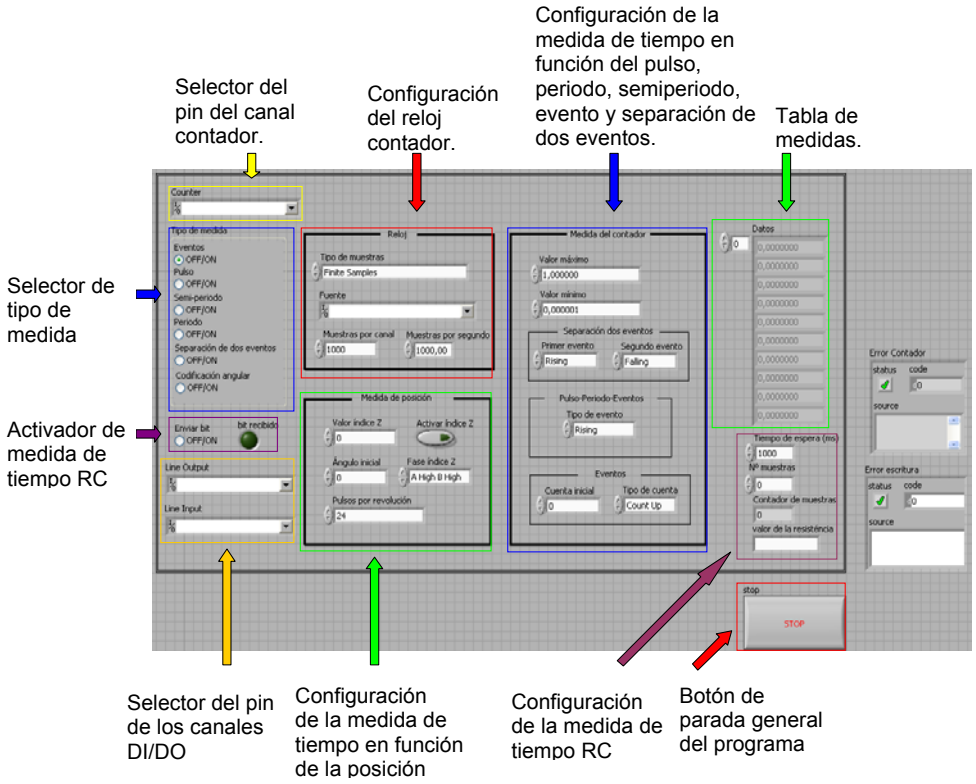

Fig. 2.26 Interfaz de control de medida de tiempo de las señales y de carga de condensadores

3. Gestión de los buses de comunicación

La última fase del proyecto es controlar los sistemas de comunicación entre los sistemas analógicos y/o digitales. Para ello, se ha utilizado la tarjeta CEI – 620 de Condor Engineering. La tarjeta se compone de dos conectores. Cada uno de ellos dispone de dieciséis canales de recepción, dieciséis canales de transmisión, ocho entradas discretas y otros ocho de salida. Además, cuenta con la opción de seleccionar la velocidad del bus. Esta puede ser de 12,5 kbits/s o de 100 kbits/s. Si se desea saber más especificaciones sobre esta tarjeta ver el Anexo I.

Por otra parte, los buses de comunicación con los que se va a trabajar son los buses aviónicos. Estos tienen la característica de ser apantallados, trenzados y diferenciales. El apantallamiento de los cables permite aislar los cables de cualquier interferencia electromagnética externa, mientras que el trenzado disminuye las interferencias capacitivas que puedan producirse entre los dos cables. Finalmente, las señales que se envían a través de los cables son diferenciales porque de esta forma se elimina cualquier error de referencia, ya que esta es cero. Otras características que pueden apreciarse son: la unidireccionalidad y el empleo de la modulación bipolar de retorno a cero. Esta última permite disponer en la comunicación de un sistema autosincronizado y de base de tiempos automatizada. El número de canales se encuentra determinado en el Anexo I, después de las características de la placa CEI-620. Así mismo también se indica a qué corresponde a cada canal.

La finalidad de este capítulo es el de crear un programa que permita gestionar dichas comunicaciones. Puesto que no se dispone en el laboratorio de ningún instrumento aviónico que utilice como sistema comunicación los buses ARINC 429, se ha visto necesario crear otro programa que permita visualizar la transmisión de los datos.

Antes de comenzar a explicar la programación, es necesario saber los mecanismos de transmisión de los buses ARINC 429. Por esta razón se ha optado por resumir las normativas que lo definen.

3.1. Normativa ARINC 429

En la normativa se encuentran diferenciadas dos tipos de datos a enviar. Los datos que contienen información para el sensor, la fuente u otros dispositivos conectados, y los datos que únicamente sirven para establecer las comunicaciones entre dos elementos. Mientras que los datos del primer tipo siguen siempre una estructura establecida, los datos del segundo tipo tienen poca o ninguna relación. Sin embargo en ambos casos se cumple que los datos siempre se envían a través de palabras de treinta y dos bits.

3.1.1. Datos de información

Para el primer tipo, existen varios tipos de códigos (datos) a enviar. Los más destacados son BCD, BNR, Datos Discretos, Alfabeto ISO y Datos de Mantenimiento y Reconocimiento. Más adelante se explicará en detalle los tres primeros que son los que se utilizan en este proyecto. El cuarto sólo se citará algunas de las características más destacadas, mientras que el quinto al no ser de interés en este proyecto, no se comentará. En cualquier caso, todos ellos cumplen la siguiente estructura:

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	1
P	SSM		Data																	Pad		SDI		LABEL	
0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	103	

Fig. 3.1 Estructura genérica de una palabra que contiene información.

Los ocho primeros bits corresponden a la identificación de la dirección. Su función básica es indicar a donde se transmite información.

Los bits nueve y diez muestran información correspondiente a la identificación fuente/destino (SDI). Normalmente suele utilizarse para transmitir a varios receptores. Puede decirse que es otro elemento que ayuda a identificar el destino final de la información. En algunos casos también se utiliza para la transmisión de datos. Si se utilizan para transmisiones múltiples en los buses ARINC 429, en ningún momento debe enviarse información a más de veinte receptores. Cabe decir que al introducir la identificación cero se llama a todos los elementos que tengan la misma dirección.

Los datos que aprovechará realmente el destinatario se encontrará en los bits entre las posiciones del 11 al 29. La forma que tendrá este campo variará en función del tipo de datos a enviar.

En las posiciones 30 y 31 de cada palabra, se muestra información de la señal y del estado de la matriz (SSM). En concreto, este campo muestra las condiciones del hadware equipado, el modo operacional, o validar el contenido de los datos.

El bit 32 corresponde al bit de paridad. Su utilidad es la de indicar si la transmisión contiene errores o no. Existen tres tipos de paridad, la paridad *Odd*, la *Even* y por último cuando no hay paridad en el envío.

Para el caso de la paridad *Odd*, el bit de paridad es uno cuando el número de bits que hay dentro de la palabra sumen una cifra impar.

Una vez visto todos los elementos que hay dentro de una palabra se dará paso a las explicaciones referidas a las características de los tipos de datos, que pueden enviarse.

BCD (Binary – Coded - Decimal)

Se utiliza normalmente para enviar datos numéricos, aunque también puede transmitir caracteres. Los bits de datos se dividen en cinco semioctetos, a excepción del primer grupo que se compone de tres bits (29, 28, 27). La representación numérica en cada semiocteto se realiza de forma binaria.

Ejemplo:

Distancia DME 25.786 MN

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	1
P	SSM		0	1	0	0	1	0	1	0	1	1	1	1	0	0	0	0	1	1	0	SDI		LABEL	
	0	0	2			5				7				8				6							

Fig. 3.2 Estructura de una palabra tipo BCD.

En la tabla que se muestra a continuación, se indica como se asignan números decimales a números binarios.

Tabla 3-1 Paso de números decimales a binarios.

Dato numérico	Dato binario
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

A veces, la información que se envía no siempre es positiva, como por ejemplo el ángulo de ataque del ala. Esta puede ser positiva o negativa. En este caso el signo se verá reflejado en el campo SSM.

En el siguiente recuadro se muestra los estados del campo SSM según los bits recibidos.

Tabla 3-2 Estado de la palabra BCD según los bits 30 y 31.

Bits	Estado
0 0	Más, Norte, Este, Derecha
0 1	Sin datos
1 0	Test Funcional
1 1	Menos, Sur, oeste

BNR (Binary)

Al igual que en el caso anterior, se usa para enviar datos numéricos. También se estructuran en semioctetos. Sin embargo, la representación de los datos numéricos en los bits es diferente. A cada bit se le asigna un valor con una relación con el bit anterior de un medio. Esto quiere decir que el bit posterior tiene el mismo valor que el otro pero dividido entre dos. En este caso, el bit más significativo es el que se encuentra en la posición veintiocho. Su valor es de un medio. La conclusión que se puede extraer es que los datos se configuran en forma de serie numérica tal y como se muestra a continuación. Donde k es la constante de cada dirección, n la posición del bit y la M el valor numérico a representar.

$$M = \sum_{n=1}^{n=18} k * \frac{1}{2^n} \qquad \textbf{(3.1)}$$

A continuación se expone un ejemplo para clarificar los valores que adquieren los bits:

Tabla 3-3 Ejemplo de asignación de valor numérico para cada bit.

28	27	26	25	24	23	22	21	20
1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512

Una vez explicado que valores tiene cada bit según su posición, para poder obtener la cifra deseada, se debe introducir unos y ceros en las posiciones adecuadas de tal forma que la suma de las posiciones multiplicadas por un factor que se determina por el tipo de información a transmitir, sea igual a dicha cifra.

Ejemplo:

Ground Speed: 650 nudos.

Constate: 4096

Configuración de los bits:

Tabla 3-4 Ejemplo de un campo de datos correspondiente a una palabra.

29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
0	0	0	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0

1/8x4096 + 1/32x4096 + 1/512x4096 + 1/2048x4096 →

512 + 128 + 8 + 2 = 650

Además de variar en la representación de los datos también discierne a la hora de indicar el signo y las funcionalidades de los bits 30 y 31. Para indicar si un número es negativo se envía bit uno en la posición 29, mientras que si es positivo se transmite cero. Cuanto se implementa uno en la posición anterior, la serie numérica cambia a la siguiente que se muestra abajo.

$$M = -k + \sum_{n=1}^{n=18} k * \frac{1}{2^n} \quad \textbf{(3.2)}$$

En la tabla inferior se muestran las funcionalidades de los bits 30 y 31 según sus datos binarios.

Tabla 3-5 Estado de la palabra BNR según los bits 30 y 31.

Bits	Estado
0 0	Fallo
0 1	Sin datos
1 0	Test Funcional
1 1	Operación normal

Datos Discretos

Los bits de datos que se envían ya no representan números sino funcionalidades a controlar. Dependiendo de la etiqueta, es decir, a donde se transmite los datos, cada posición de bit que pertenezca al campo de datos, representará un elemento a controlar u otro. Además, es posible que en cada funcionalidad, el envió de uno o de cero no signifique siempre lo mismo. En algunos casos significa correcto o fallo, mientras que en otros activo o inactivo.

Para que el concepto quede claro, a continuación se muestra una tabla con todos los bits de la dirección 5 pertenecientes al campo de datos.

Tabla 3-6 Ejemplo de elementos dentro de una palabra discreta

Bit	Función	1	0
11	PAD		X
12	PAD		X
13	Failure to clear serial interrupt	Fail	Pass
14	ARINC recived fail	Fail	Pass
15	PROM checksum fail	Fail	Pass
16	User RAM fail	Fail	Pass
17	NV RAM address fail	Fail	Pass

18	NV RAM bit fail	Fail	Pass
19	RTC fail	Fail	Pass
20	Microprocessor fail	Fail	Pass
21	Battery low	Fail	Pass
22	NV RAM corrupt	Fail	Pass
23	Not used		
24	Not used		
25	Not used		
26	Interrogated activated		
27	Erase activated	Activated	Non-Activated
28	Bit activated	Activated	Non-Activated

Las funcionalidades para el caso de Datos Discretos según los valores que adquieren los bits 30 y 31 se muestran en la siguiente tabla.

Tabla 3-7 Estado de la palabra según los valores de los bits 30 y 31.

Bits	Estado
0 0	Verificación de datos, Operación normal
0 1	Sin datos
1 0	Test Funcional
1 1	Fallo

Alfabeto ISO

Los datos que se envían son caracteres. Los valores binarios de cada carácter vienen determinados en las normativas ARINC 429, Attachment 5, primera parte.

Hay cuatro tipos de palabras que se transmiten. Dos de ellas corresponden a palabras iniciales, la tercera a palabras intermedias y la cuarta a palabras finales. En todas ellas el campo de datos los bits quedan agrupados en tres grupos. El primer grupo que es el más significativo, se encuentran los bits entre las posiciones 29 y 23. El segundo grupo se compone de los bits comprendidos entre el 22 y 17. El último grupo se hayan los bits situados entre los lugares 16 y 9.

La primera palabra inicial consta en su campo de datos, una palabra de protocolo de comunicación entre la fuente y el sistema, la dirección del sistema y de otra palabra cuya función es la de contar las palabras de 32 bits que se mandan.

A continuación se muestra su estructura:

P	SSM (01)	"STX"	UNIT ADDRESS	WORD COUNT	LABEL (357)
32	31 30	29 23	22 17	16 BNR EQUIV. 9	8 1

Fig. 3.3 Estructura de la primera palabra inicial tipo alfabeto ISO.

La segunda palabra inicial es similar a la primera pero con la diferencia de que esta segunda no transmite datos de dirección entre las posiciones 17 y 22. En su lugar transmite bit cero en todas las posiciones de los bits donde se antes se encontraba los datos de dirección.

P	SSM (01)	"STX"	SPARES (Zeroes)	WORD COUNT	LABEL (356)
32	31 30	29 23	22 17	16 BNR EQUIV. 9	8 1

Fig. 3.4 Estructura de la segunda palabra inicial tipo alfabeto ISO.

En las palabras intermediarias únicamente se mandan caracteres en el campo de datos tal y como se visualiza en la figura siguiente.

P	SSM (00)	"DATA CH #3"	DATA CH #2	DATA CH #1	LABEL (356, 357)
32	31 30	29 P 23	22 L 16	15 A 9	8 1

Fig. 3.5 Estructura de una palabra intermedia tipo alfabeto ISO.

Finalmente, las palabras finales se caracterizan por el hecho de que no se transmita ningún carácter en uno o dos de los tres grupos. En el grupo de bits donde no se transmite ningún carácter, se rellena con ceros. En la figura que viene a continuación se muestra un ejemplo.

P	SSM (10)	"DATA CH #3"	DATA CH #2	DATA CH #1	LABEL (356, 357)
32	31 30	29 (BNR ZEROES) 23	22 A 16	15 H 9	8 1

Fig. 3.6 Estructura de la palabra final tipo alfabeto ISO.

3.1.2. Datos de protocolo.

Para los sistemas que mandan o reciben caracteres, las funciones básicas de estos tipos de datos son establecer contacto con los sistemas receptores, comprobar en todo momento que no falte ningún elemento de la palabra o ninguna palabra del mensaje enviado y asegurarse que la transmisión entre fuente y emisor carezca de errores. Sin embargo, los dispositivos que transmiten datos numéricos sólo necesitan establecer el contacto, incluso en

ocasiones la información esencial se envía directamente a los dispositivos sin necesidad de emitir datos de protocolo. Puesto que en este proyecto sólo se centra en datos numéricos, se obviará el protocolo para el envío de caracteres y se explicará el protocolo relacionado con el envío de números.

Cuando una fuente intenta contactar con un dispositivo, debe establecerse el protocolo de comunicación "Bit-Oriented Protocol Determination". Este protocolo consiste en que la fuente debe enviar antes que nada la palabra ALO para determinar si el receptor está disponible. Si no recibe ninguna respuesta, esperará entre 200 y 250 milisegundos, tanto si la transmisión es de 100 kbytes/s o de 12,5 kbytes/s., para a volver a repetir la misma palabra.

Si al tercer envío de la palabra ALO, la fuente no recibe ninguna respuesta, declarará a al receptor en concreto como no disponible o que no dispone del protocolo de transmisión "Bit-Oriented Protocol Determination".

Cuando el receptor se encuentra disponible, manda la palabra ARL a la fuente con un retraso máximo de 180 milisegundos, tanto para la velocidad de 100 kbytes/s, como la de 12,5 k bytes/s.

Una vez que las dos palabras se han enviado y recibido satisfactoriamente, la fuente puede proceder a enviar los datos numéricos.

A continuación se muestra la estructura de la palabra ALO.

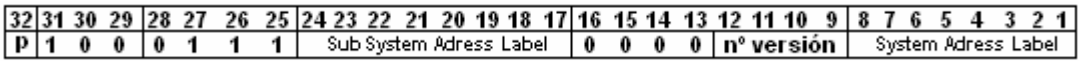

Fig. 3.7 Estructura correspondiente a la palabra ALO

Al igual que en la palabra para enviar datos, los ocho primeros bits indican la dirección a la que son transmitidos los datos. Los bits del 9 al 12 se usan para indicar la versión del protocolo que se usa. En concreto hay cinco versiones, aunque en este caso con la primera versión es suficiente. Entre las posiciones 17 y 24 de la palabra ALO, vuelve a introducirse la información sobre la dirección. Los datos introducidos entre los bits 25 y 28, hacen referencia a la palabra protocolo en si, es decir, que en estas posiciones se introduce la palabra ALO. Los tres siguientes bits indican que tipo de palabra se envía. El último bit es el de paridad.

En cuadro que viene a continuación se muestra las posibilidades que pueden darse en los bits de las posiciones 29 a 31.

Tabla 3-8

31	30	29	WORD TYPE
0	0	0	Full Binary Data Word

0	0	1	Partial Binary Data Word
0	1	0	Start of Frame – Version 3
0	1	1	End of Frame – Version 3
1	0	0	Protocol Word
1	0	1	Solo Word
1	1	0	Start of Transmision – Version 1
1	1	1	End of Transmisión – Version 1

En la siguiente figura se visualiza la estructura de la palabra ARL.

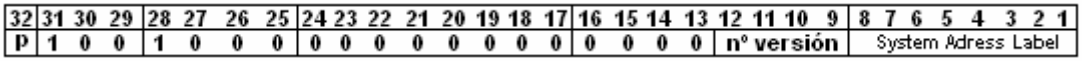

Fig. 3.8 Estructura correspondiente a la palabra ARL.

Al igual que en la palabra ALO, en la palabra ARL, los ocho primeros bits se utilizan para indicar la dirección del mensaje mientras que los cuatro siguientes muestran la versión del protocolo. Sin embargo, a diferencia que la palabra ALO, en esta palabra no se vuelve a introducir la dirección entre los bits 17 y 24. En este caso se rellenan todas las posiciones con ceros. Además, entre los bits 25 y 28 se debe introducir los datos adecuados para indicar que es la palabra ARL. El resto de los bits tienen la misma función y se colocan igual a la palabra anterior.

3.2. Gestión de los buses ARINC 429.

Primero se describirá los dos programas principales. En el Anexo II se explicará un pequeño programa que permitirá guardar y cargar de forma automática las estructuras de los diferentes tipos de datos transmitidos. Además, en el mismo anexo se comentará de forma más detallada como implementar una función que pase de números decimales a números binarios y viceversa para la estructura BCD y BNR.

3.2.1. Programa de transmisión de datos

En este apartado se quiere crear un módulo que permita configurar tanto para los canales de transmisión como los de recepción, la velocidad del bus, la paridad, el canal y el tipo de tarjeta. También se deberá tener la opción de enviar los datos de forma numérica y en una palabra de 32 bits.

Otra funcionalidad que se realizará es la de introducir nuevas estructuras de palabras según las direcciones. Adicionalmente, el módulo deberá poder guardar o cargar estructuras de palabras.

Independientemente de las funcionalidades que realice, también ha de ser capaz de realizar el protocolo de comunicación entre la fuente y los sistemas de

comunicación. Ello incluye la transmisión de la palabra ALO y la recepción de la palabra ARL.

3.2.1.1. Driver realizado

Las funciones utilizadas para esta ocasión son diferentes a las realizadas para la placa NI PXI-6221, ya que la placa para el control de las comunicaciones pertenece a la empresa “Condor Engineering”, mientras que el programa utilizado sólo permite configurar hadware perteneciente a la empresa NI. De esta forma puede utilizar el mismo programa para configurar los dispositivos.

A continuación se procederá a explicar la implementación de este programa.

La primera aplicación con la que comienza el programa es “ar_board_init”. Esta función permite inicializar la tarje. El segundo elemento que se encuentra es la "ar_clear_xmitdb”, cuya función es limpiar todos los mensajes que se estén transmitiendo entre el sistema PXI y el portátil. A continuación, está la aplicación “ar_timetag_control”. Esta aplicación permite controlar los tiempos de lectura del buffer de cada palabra recibida o transmitida. Después, se configuran los canales de transmisión y recepción a través de la función “ar_config_chnl”. Puesto que cuando se realice la comunicación de protocolo el programa transmisor también va a necesitar obtener una respuesta, es necesario introducir dos veces la misma función. Una de ella se configurará para que sea transmisor y otra para que sea receptor. La aplicación que le precede es la inicialización las tablas de conversión, “ar_init_cs_conv”. El siguiente elemento, es una estructura de tipo caso (“Case”). Se ejecuta la parte falsa si la variable booleana a la que esta conectada le da un valor falso. En caso contrario, realiza la parte cierta.

Dentro de la parte falsa, la estructura únicamente carga en la librería vinculada a las funciones que necesiten introducirse las direcciones, las identificaciones del equipo y las matrices de estado, “ar_build_custm_conv”. En la parte cierta, además de introducir los datos para nuevas estructuras de palabras, se cargan ficheros a través de la función “Guardar Librerías”.

Para terminar con la inicialización y configuración de la tarjeta, la siguiente función a implementar es “ar_def_hwmsg_raw”, que permite configurar el mensaje de transmisión inicial.

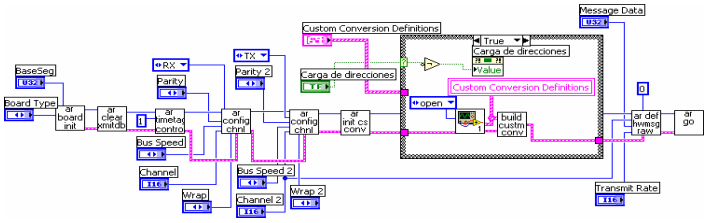

Fig. 3.9 Esquema de la inicialización de la tarjeta y sus canales.

Para activar la transmisión de los datos se implementa a continuación de las funciones anteriores la aplicación “ar_go”. Una vez activada la señal se procederá a transmitir los datos.

Después de la función “ar_go”, se introduce una estructura de secuencia (“Stacked Sequence Structure”). En la primera secuencia se transmitirá la palabra ALO. A continuación, se ejecuta un bucle en el que se detendrá cuando se exceda el tiempo establecido por la normativa ARINC 429 o el número de repeticiones. Para transmitir la palabra ALO, se implementa la función “ar_mod_hdmsg_raw”. A la hora de transmitir la palabra hay que tener en cuenta que en su estructura hay un elemento que varia, la dirección. Es por ello que se ha decidido implementar dicha palabra en cuatro paquetes. Cada paquete es una variable de ocho bits, en el cual, dos de ellos se implementarán mediante números binarios. Estos dos paquetes corresponden a estructuras predefinidas en la palabra ALO. Los otros dos, que corresponden a la dirección, se introducirán con números octales. Después, se unen todos los paquetes en el mismo orden que aparece en la Fig. 3.7 a través de la función “Join Numbers”. Una vez obtenido la palabra final, se transmite a la función “ar_mod_hdmsg_raw”.

Para leer las palabras que va recibiendo, se coloca dentro del bucle la función “ar_timed_read”. Para poder comprobar que el bucle realiza la espera según las normativas, se usa la aplicación “ar_gttmr_cntl” permite contar de forma precisa el tiempo que transcurre en milisegundos. Sin embargo, tiene un problema, y es que una vez invocado dicha aplicación, las cuentas no pueden inicializarse nuevamente a cero a menos que se detenga la ejecución de la placa. Para solucionar este problema se coloca la misma función antes de entrar al bucle y otra dentro del bucle, después de la función “ar_timed_read”. A continuación, se le resta al valor obtenido anteriormente, el tiempo adquirido antes de entrar en el bucle.

Al contrario que se ha hecho a la hora de enviar la palabra ALO, la palabra que se recibe se divide en tres paquetes con la función “Split Number”. Dos de ellos son variables de ocho bits mientras que el último es de dieciséis bits. A continuación, se compara dicho paquetes con la parte de la estructura ARL correspondiente. Si la comunicación es nula, se enciende la booleana de Error. Para limpiar los mensajes guardados en la memoria virtual, se coloca nuevamente las funciones "ar_clear_xmitdb” y “ar_init_lb_hist_db”.

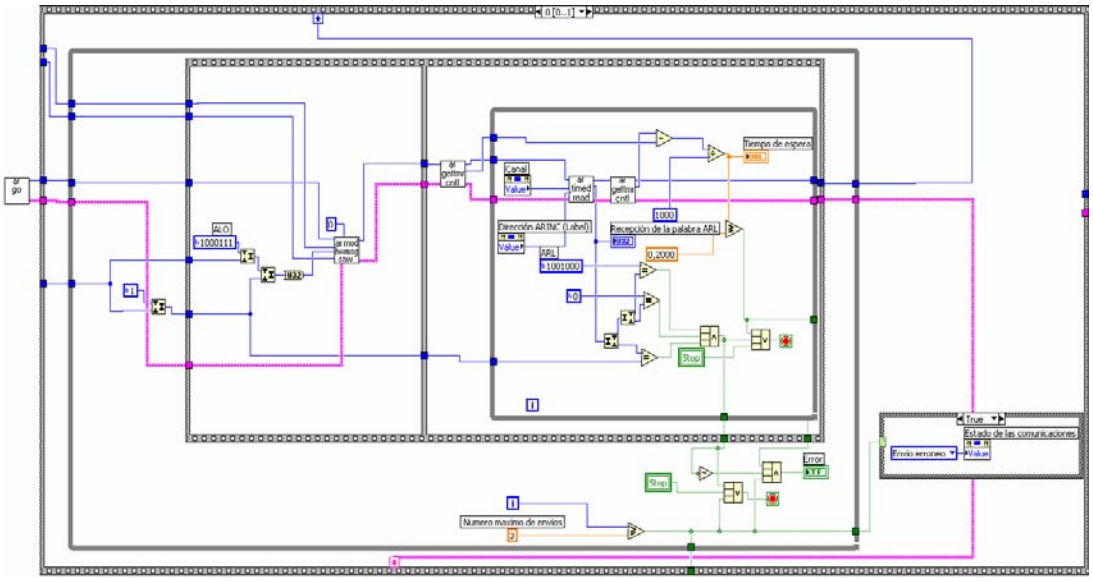

Fig. 3.10 Estructura de protocolo de transmisión-recepción.

En la segunda secuencia, si la booleana es cero, permitirá ejecutar el envío de los datos numéricos.

La transmisión de los datos puede realizarse de dos maneras, o bien introduciendo todos los datos necesarios para la transmisión al mismo tiempo, o hacerlo por separado. Para representar esta opción, dentro de la secuencia se encuentra implementado una estructura de Caso (“Case”). Dentro del “Case” cero se encuentra la función “ar_def_hwmsg_raw”, que permite transmitir mediante la primera opción. En el “Case” uno, se haya la aplicación “ar_def_hwmsg_eng”, que permite colocar los datos por separado, es decir, en una variable se introduciría los datos numéricos, en otra variable la dirección, en otra la identificación del equipo (ID), en otra variable la matriz de estado y en otra la identificación fuente destino. Debido que en las normativas no se especifica con claridad cómo se selecciona cada ID que hay dentro de cada dirección, se ha optado por transmitir el valor cero. Cuando se transmite dicho valor en la ID, significa que llama a todos los elementos que se encuentren dentro de una dirección en concreto. Para poder introducir datos sin ocupar demasiado espacio de programación se ha optado por agrupar todos los elementos anteriormente citados, incluido la variable que permite colocar directamente palabras de 32 bits, en un cluster denominado Canal transmisión cuya estructura se muestra a continuación.

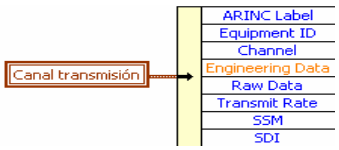

Fig. 3.11 Variables dentro del cluster “Canal transmisión”.

Una vez escogido en que modo se transmitirá y se hayan introducidos los datos en la variable Canal transmisión, se mandará nuevamente los mensaje. Para ello es necesario volver a situar de nuevo la aplicación “ar_go”. A continuación, le sigue un bucle que permite mandar los datos deseados hasta que se detenga el programa o se cambie de dirección.

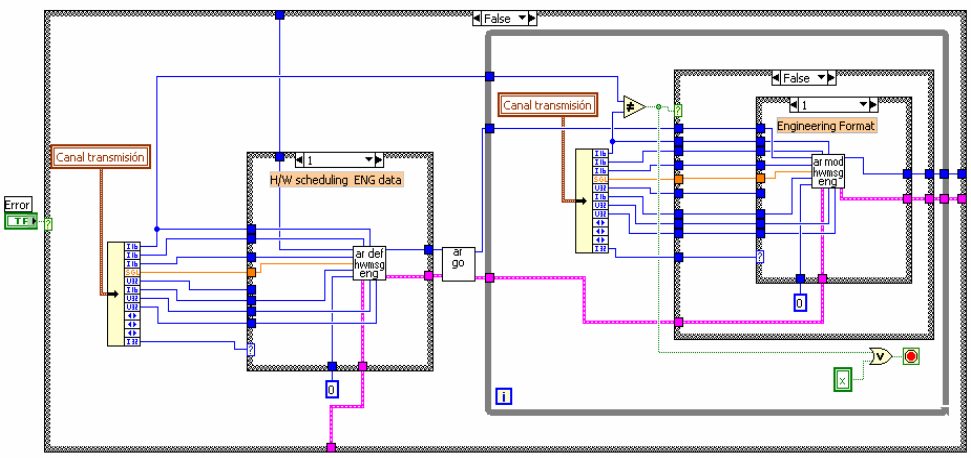

Fig. 3.12 Esquema de transmisión de datos.

Cuando el programa se detenga, se ejecutará nuevamente las funciones "ar_clear_xmitdb” y “ar_init_lb_hist_db”.

3.2.1.2. Interfaz del usuario

En este apartado se comentará todos los botones y controladores disponibles en la interfaz. Así mismo, se explicara el funcionamiento de cada uno de ellos y de todo el programa completo. La presentación de la pantalla se muestra en la Fig. 3.13.

La pantalla en el que el usuario trabajará, se visualiza en la parte superior el controlador que permite escoger el tipo de placa a controlar. A su derecha aparece el controlador que controla el inicio de la memoria de la placa. Al final a la derecha se muestra el indicador donde se comprueba si la palabra ARL se ha adquirido correctamente.

En la parte inferior se observan dos clusters, uno corresponde a Canal transmisión y el otro a Canal receptor. En el primero se pueden modificar parámetros relacionados con el canal. Dichos parámetros son, la dirección, la identificación del equipo, el canal, la paridad, la velocidad del Bus, el wrap, indicar si lo que se quiere transmitir por separado o en conjunto, modificar numéricamente los datos decimales, modificar directamente una palabra a transmitir, la velocidad a la que se transmite los bits, el estado de la matriz, y la identificación de la fuente / destino.

EL cluster situado más a la derecha corresponde a Canal receptor. En él es posible modificar el wrap, el bit de paridad, la velocidad del bus y el canal.

En la misma columna donde se encuentra el cluster Canal receptor, en la parte de abajo se puede visualizar el tiempo que cuenta el programa en cada iteración. Más abajo también se observa una pequeña pestaña que puede mostrar dos opciones, Envío correcto y Envío erróneo. Si el proceso de establecer comunicación con algún destinatario es fallido el programa mostrará la pestaña Envío erróneo. Si por el contrario no pasa nada, se visualizará la pestaña de Envío correcto.

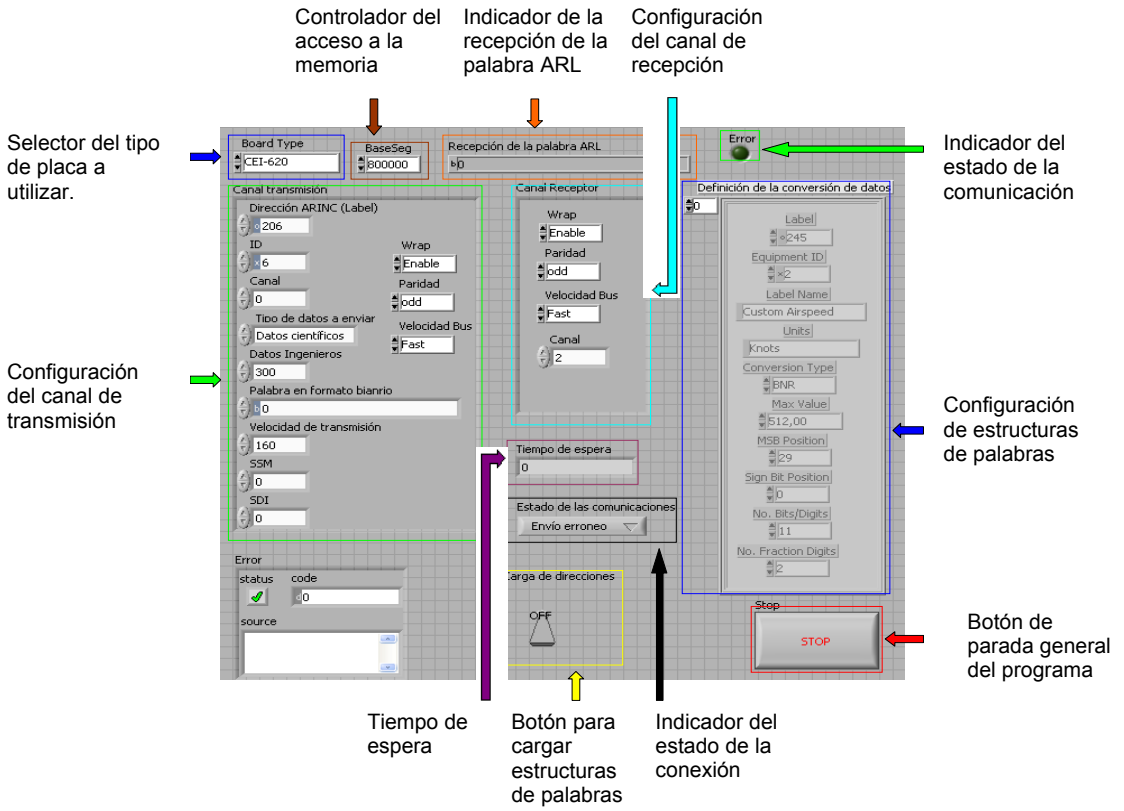

Fig. 3.13 Interfaz de control de transmisión de datos.

3.2.2. Programa para recibir datos.

La aplicación a crear en este punto tiene que tener la posibilidad de simular el protocolo de recepción de un sistema aviónico. Al igual que el módulo anterior, ha de ser capaz de controlar en la transmisión y en la recepción, la velocidad del bus, el tipo de paridad y el canal. Debe permitir seleccionar la tarjeta específica que se utiliza, el inicio de la memoria. También ha de ser capaz de cargar estructuras de palabras, ya sea manualmente o a través de la carga de un archivo. Finalmente, debe permitir visualizar los valores numéricos y mostrarlos en una gráfica.

3.2.2.1. Driver realizado

Al igual que en el apartado 3.2.1, se utilizarán las mismas aplicaciones implementadas por Condor Engineering para .crear esta aplicación. En el siguiente párrafo, se comienza a describir su esquema.

La estructura inicial es la misma que aparece en la Fig. 3.9. Lo que varia en este caso es la implementación de las dos secuencias de la estructura de secuencia que le sigue a continuación.

En la primera secuencia, al igual que en el programa anterior. Las únicas variaciones que aparecen son dos. La primera es que el programa espera primero a recibir la palabra protocolo, que en este caso es ALO, y después transmite la palabra ARL. La segunda es que se coloca en la secuencia donde se transmite la palabra ARL la aplicación “Wait Until Next ms Múltiple” con una espera de 150 milisegundo. Básicamente permite que el programa transmisor reciba la señal sincronizada.

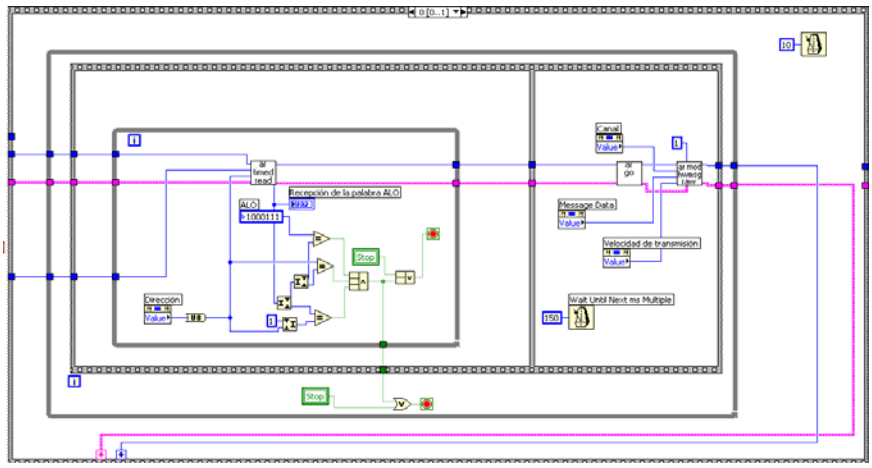

Fig. 3.14 Esquema de protocolo recepción-transmisión.

En la segunda secuencia, recibe datos numéricos. Para poder adquirir los datos continuamente, se implementa dentro de la secuencia un bucle que se ejecutará hasta que se detenga el programa, o cambie la dirección o identificación del equipo. A su vez, dentro del bucle se introduce la función “ar_timed_read”. El siguiente elemento utilizado es una estructura de caso (“Case”) que evaluará si se ha cambiado de dirección o identificación. En caso cierto se ejecutará la función “ar_arinc_toascii”. Esta aplicación permite transformar las palabras de 32 bits números decimales siempre y cuando dicha estructura de palabra se encuentre introducida en la librería correspondiente. Finalmente, los datos son graficados a través del indicador “waveform Chart” y mostrados numéricamente mediante la variable Muestra numérica.

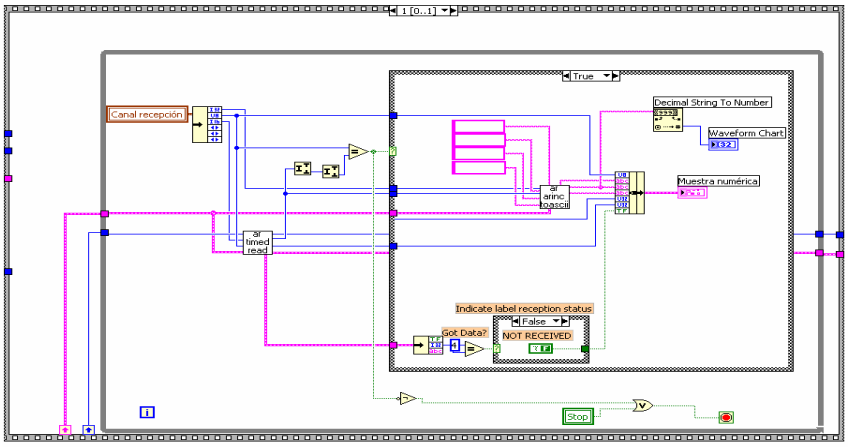

Fig. 3.15 Esquema de recepción de datos.

3.2.2.2. Interfaz del usuario

En este punto, al igual que en los puntos 1.1.2, 1.2.2, 2.1.2, 2.2.2 y 3.2.1.2, se pretende explicar todos los elementos que se compone la parte del programa que esta de cara al usuario. Su presentación puede visualizarse en la Fig. 3.16.

En la interfaz se muestra en la parte superior los controladores que permiten elegir por una parte el tipo de placa que se esta utilizando, y por la otra, el inicio de la posición de la memoria en la que el sistema PXI accede para adquirir o transmitir los mensajes. A continuación, en la parte inferior aparece un indicador que permite visualizar si la palabra protocolo ALO ha llegado satisfactoriamente.

En el cluster Canal recepción, que se visualiza por debajo del indicador anterior, permite configurar el canal receptor. En su interior se puede apreciar la variable que controla la dirección del mensaje, la que modifica la velocidad de transmisión, la que varia el tipo de bit de paridad, el que controla el canal por donde se recibe, y el que modifica la ID.

El siguiente cluster situado a la derecha del anterior elemento, corresponde a Canal transmisión. En esta ocasión, puede modificarse la transmisión de los bits, pero en cambio, es imposible modificar la dirección y la identificación del equipo.

A la derecha extrema, se encuentra el cluster Muestra numérica. Dentro de esta variable, aparecen la parte superior la dirección representada de forma octal y el nombre al que corresponde. Más abajo se visualiza los datos numéricos representados decimalmente y las unidades respectivas. A continuación se muestra en la zona inferior la velocidad a la que se recibe los datos, y la palabra de 32 bits.

Finalmente, en la parte más baja de la interfaz se muestra una gráfica, otro cluster para mostrar posibles errores de transmisión y el botón de parada general del programa.

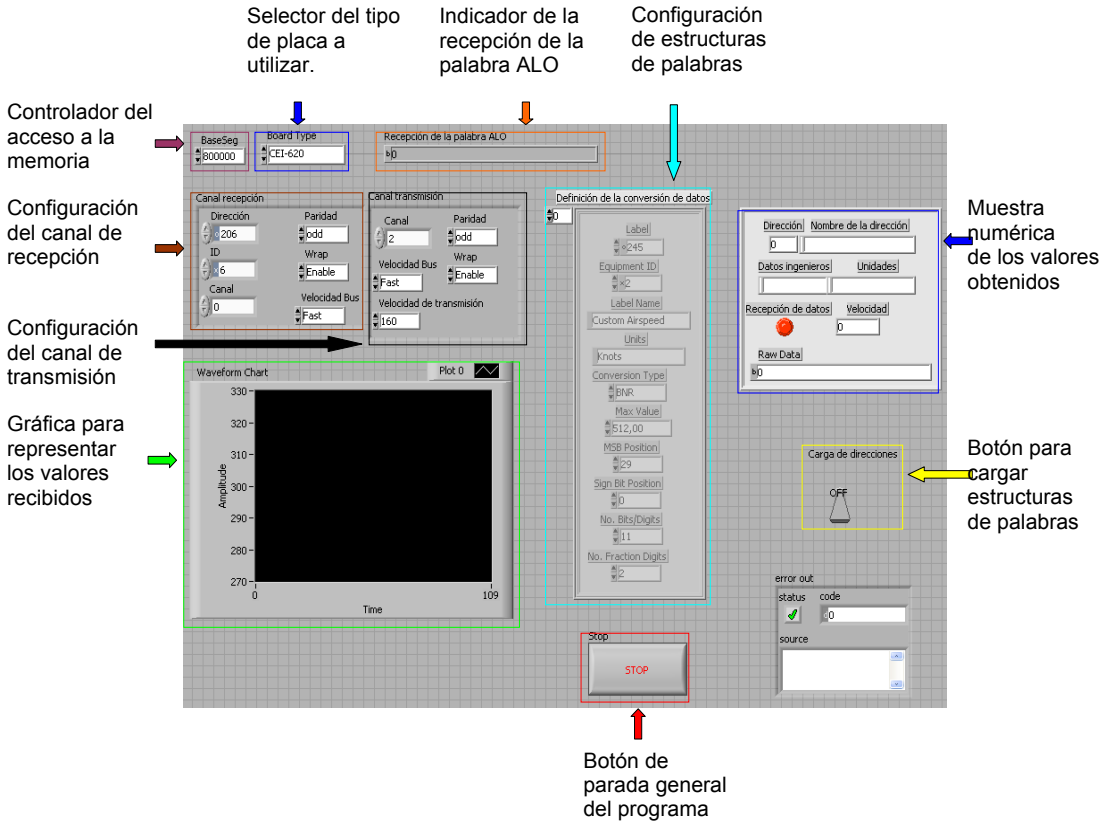

Fig. 3.16 Interfaz de control de recepción de datos.

3.2.3. Resultados obtenidos

Para finalizar con el tercer y último capítulo, se muestran los resultados obtenidos por el osciloscopio de la transmisión y recepción de los datos por el bus de comunicaciones.

Si se comienza a mirar los bits en la Fig. 3.17 por la izquierda, se puede observar que los ocho primeros bits hacen referencia a la dirección número 103. Para comprobarlo en las normativas ARINC 429, hay que tener en cuenta que el primer bit corresponde a la octava posición de la dirección, mientras que el octavo bit representa la primera. Los cuatro siguientes bits que representan la versión del protocolo, se visualiza que el primer bit que corresponde a la novena posición de la palabra, es uno, y por tanto la versión que se ha utilizado es la primera. Los cuatro bits que le preceden son los que simplemente se introducen como cero, mientras que los ocho bits que vienen después vuelven a corresponder a la dirección. Los cuatro siguientes representan la palabra ALO y los tres últimos al tipo de palabra.

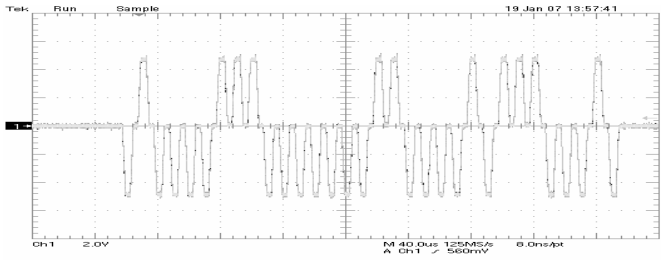

Fig. 3.17 Palabra ALO de la dirección 103.

La respuesta que se obtiene del programa receptor es la deseada, ya que como se muestra en la Fig. 3.18, los ocho primeros bits representan a la dirección, mientras que los cuatro siguientes indican al transmisor la versión de protocolo. Los siete últimos bits sin contar con el bit del final, corresponden al tipo de palabra y a la palabra ARL. Los bits que se encuentran en las posiciones del trece al veinticuatro, son los ceros definidos por la normativa.

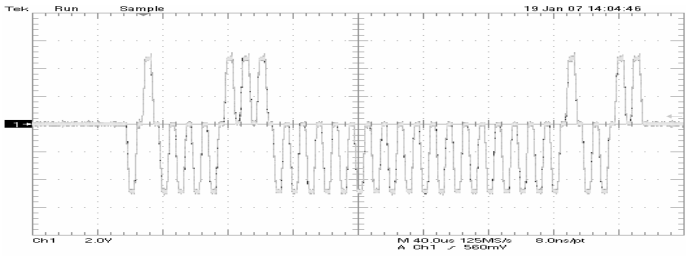

Fig. 3.18 Palabra ARL de la dirección 103.

En la Fig. 3.19, se muestra un claro ejemplo obtenido de la transmisión de una palabra que corresponde a datos de información de la misma dirección anterior. Si se busca en las normativas el tipo de estructura que tiene esta dirección se observará que es de tipo BNR. El número enviado en este caso es el 733.

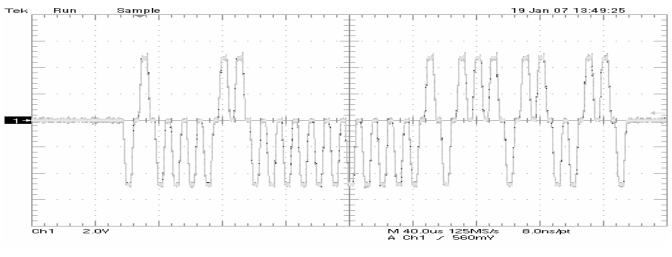

Fig. 3.19 Palabra con el valor 733 de la dirección 103.

En la Fig. 3.20, el número que se representa es el mismo pero con signo negativo. Esta variación se visualiza en la posición veintinueve de la palabra con el envío de bit uno.

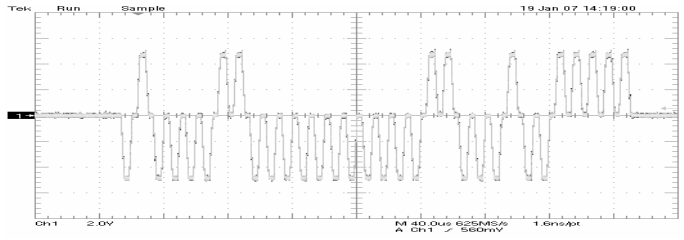

Fig. 3.20 Palabra con el valor –733 de la dirección 103.

Para mostrar un ejemplo visual de la estructura BCD recibido en un osciloscopio, se cambia la dirección de 103 a la 201. En las normativas se especifica que la dirección 201, corresponde a este tipo de estructura. El número representado para esta ocasión es 245,89. La estructura final es la que se muestra en la Fig. 3.21.

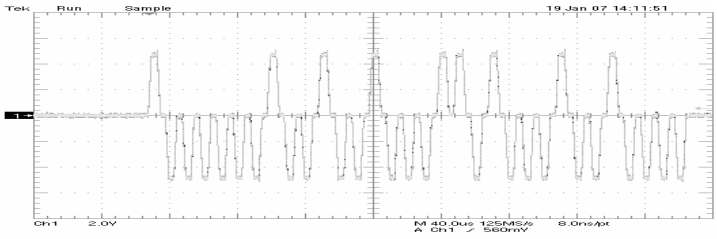

Fig. 3.21 Palabra con el valor 245,89 de la dirección 201.

En la imagen que viene a continuación representa el número anterior pero con signo negativo.

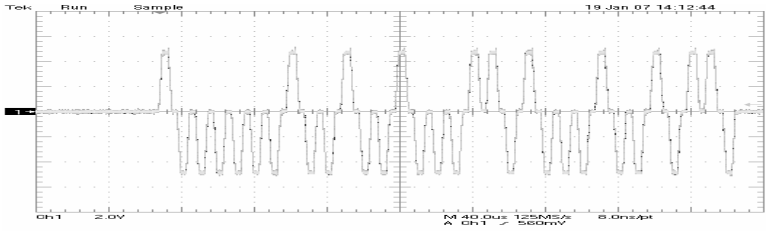

Fig. 3.22 Palabra con el valor –245,89 de la dirección 201.

Se observa que el penúltimo y antepenúltimo bit, que representan la SSM, que los dos son unos. Si se revisa la estructura BCD descrita en el apartado 3.1.1, se visualiza en la Tabla 3-2 que los dos unos corresponden al signo negativo.

4. Conclusiones

Los programas realizados en este proyecto permiten gestionar perfectamente los sistemas de adquisición y excitación tanto analógicas como digitales. Cabe destacar que al ser realizado sobre la plataforma NI DQAmx pueden ser utilizados para cualquier otra tarjeta de NI, ya sea de esta plataforma PXI o en cualquier otra plataforma de instrumentación virtual. Así mismo, su realización modular permite no tan sólo la utilización de los programas globales, sino también de las diferentes funciones implementadas.

Todos los programas han sido comprobados experimentalmente, validando las especificaciones marcadas al inicio del proyecto.

Además del control de estos sistemas analógicos y digitales se ha implementado programas para gestionar los contadores de 32 bits en la tarjeta NI PXI-6221. Estas funciones han permitido realizar de forma novedosa interfaces directas entre sensores resistivos y dichos contadores, dando pie a un novedoso sistema de adquisición de datos que permite sustituir los canales analógicos de entrada por controladores cuya implementación microelectrónica es más sencilla y económica.

Para complementar las necesidades en los sistemas de test aviónicos se ha generado los programas para gestionar un bus aviónico ARINC 429 mediante la tarjeta CEI-620 de la casa Condor. Las funciones de estos programas permiten la transmisión y recepción de datos, cargar o guardar estructuras de palabras y realizar la conversión de palabras de 32 bits en códigos BCD y BNR a números decimales y viceversa.

La aplicación de este trabajo consistiría en la realización de un programa real que incorporara todos los módulos realizados y se mejorara las prestaciones obtenidas con la placa CEI-620. Esta mejora consiste en la sincronización de la transmisión y la recepción para la velocidad lenta.

Bibliografía

[1]. http://sine.ni.com/nips/cds/view/p/lang/en/nid/14133. En esta Web se especifica todas las características importantes de la tarjeta NI PXI-6221.
[2]. Condor Engineering “Installation” Cap. 2, pp. 16-17 y “Programe Interface Library” Cap 5, pp. 71-144 en *CEI-100/CEI-200/CEI-x20 User’s Manual.*
[3]. Airlines Electronic Engineering Comité, *Normativa ARINC 429,* Parte 1 y Parte 3.
[4]. Coombs F., “Introduction to Electronic Instrument and Measurement”, Cap. 1, “Data Acquisition System” Cap. 4 y “Standard–Based Modular Instrument” Cap 40 en *Electronic instrument handbook.*

Anexo I

Características

En este apartado nos detendremos para describir las características físicas y eléctricas de todos los dispositivos que van a utilizarse.

Sistema PXI:

- Modelo: PXI-1036.
- Nº máximo de módulos: 6.
- Reloj interno: 10 Mhz.
- Retraso Star trigger en la propagación de la señal: 5 ns.
- Retraso Star trigger entre Slot – Slot: 1 ns.
- Rango de temperatura de operación: 0 – 50 ºC.
- Alimentación a A.C. o A.C./D.C.

Tarjeta NI PXI-6221:

- Nº de pins: 68.
- Nº de canales: 42.
  - 16 entradas analógicas.
  - 2 salidas analógicas.
  - 24 canales digitales, 4 de ellos pueden configurarse también como contadores.
- Canal de entrada analógica.
  - Velocidad máxima de muestreo: 250 KS/s.
  - Precisión de la sincronización: 50 ppm.
  - Resolución de la sincronización: 50 ns.
  - Tensiones de entrada disponibles: ±10 V, ±5 V, ±1 V,
  - ±0,2 V.
  - Máxima tensión de trabajo: ±11 V.
  - Tensión de protección conectada: ±25 V.
  - Tensión de protección desconectada: ±15 V.
  - Impedancia de entrada conectada: 10 Gohm en paralelo con 100 pF.
  - Impedancia de entrada desconectada: 820 ohm.
- Canal de salida analógica.
  - Velocidad máxima de subida: 833 KS/s.
  - Precisión de la sincronización:50 ppm.
  - Resolución de la sincronización: 50 ns.
  - Tensión de salida: ±10 V.
  - Intensidad de salida: ±5 mA
  - Impedancia de salida: 0,2 ohm.
  - Tensión de protección: ±25 v.
  - Intensidad de protección: 10 mA.
- Canal digital.
  - Control individual de dirección de cada canal.
  - Resistor nivel bajo: 50 ohm hasta 70 ohm.

- Tensión alta máxima de entrada: 5,25 V.
- Tensión alta mínima de entrada: 2,2 V.
- Tensión baja máxima de entrada: 0,8 V.
- Intensidad alta máxima de salida en Port 0: -24 mA.
- Intensidad alta máxima de salida en Port 1/2: -16 mA.
- Intensidad baja máxima de salida en Port 0: 24 mA.
- Intensidad baja máxima de salida en Port 1/2: 16 mA.
- Umbral positivo: 2,2 V.
- Umbral negativo: 0,8 V.
- Velocidad máxima de transmisión: 1 Mhz

➢ Canal contador.

- Resolución 32 bits
- Medidas de: Eventos, Pulsos, Semiperiodos, Periodos y de dos eventos independientes.
- Medidas de posición: X1, X2, X4 codificado en cuadratura con el canal Z, y codificación de dos pulsos.
- Aplicaciones de salida: Pulsos, Tren de pulsos con subidas dinámicas, divisor de frecuencias y tiempo de sincronización equivalente.
- Relojes base internos: 80, 20 y 0,1 Mhz.
- Relojes base externos de frecuencia: 0 hasta 20 Mhz.
- Precisión del reloj base: 50 ppm.

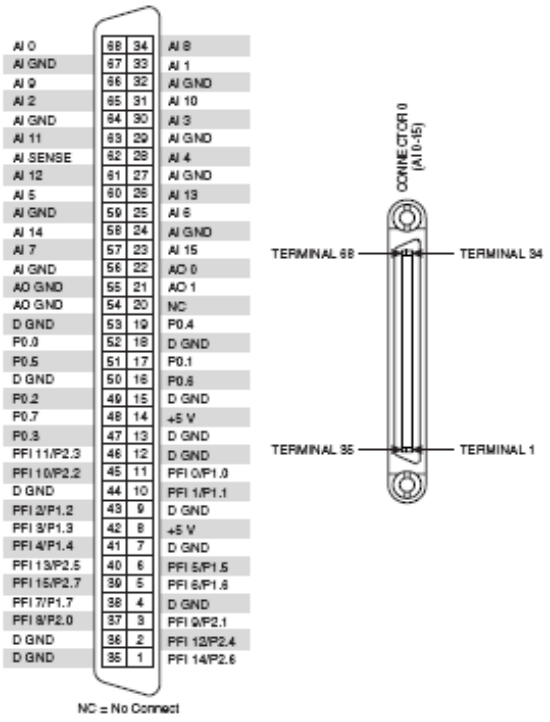

Fig. I.1 Conector NI PXI 6221

Tabla I.1 Pines correspondientes a la conexión con el contador.

Medidas	Canal Contador1	Canal Contador2
Evento	37	42
Periodo	3	41
Semiperiodo	3	41
Pulso	3	41
Dos eventos (Canal 1)	3	41
Dos eventos (Canal 2)	45	46

Tarjeta CEI-620:

- Nº de conectores: 2.
- Nº de pins: 50 por conector.
- Nº de canales de transmisión: 16 por conector.
- Nº de canales de recepción: 16 por conector.
- Nº de entradas discretas: 8 por conector.
- Nº de salidas discretas: 8 por conector.
- Velocidad de carga y descarga de datos: 12,5 khz o 100 khz.
- Paridad: odd, even o none.
- Nivel de tensión de entrada: ±6,5 hasta ±13 V.
- Niveles de umbrales: ±0,1 hasta ±13,5 V.
- Nivel de tensión de salida: 0 hasta ±10 V.
- RAM: 512 KB.
- Rango de operación de temperatura: 0ºC hasta +70ºC.

Tabla I.2 Pines correspondientes a la tarjeta.

P1 Connector			P2 Connector		
P1	D50	Signal	P2	D50	Signal
25	17	Discrete Output 7	25	17	Discrete Output 15
24	49	Discrete Output 5	24	49	Discrete Output 13
23	32	Discrete Output 3	23	32	Discrete Output 11
22	15	Discrete Output 1	22	15	Discrete Output 9
21	47	Discrete Input 7	21	47	Discrete Input 15
20	30	Discrete Input 5	20	30	Discrete Input 13
19	13	Discrete Input 3	19	13	Discrete Input 11
18	45	Discrete Input 1	18	45	Discrete Input 9
17	28	GND	17	28	GND
16	11	RX 8 A	16	11	RX 16 A
15	43	RX 7 A	15	43	RX 15 A
14	26	RX 6 A	14	26	RX 14 A
13	9	RX 5 A	13	9	RX 13 A
12	41	RX 4 A	12	41	RX 12 A
11	24	RX 3 A	11	24	RX 11 A
10	7	RX 2 A	10	7	RX 10 A
9	39	RX 1 A	9	39	RX 9 A
8	22	TX 8 A	8	22	TX 16 A

7	5	TX 7 A	7	5	TX 15 A
6	37	TX 6 A	6	37	TX 14 A
5	20	TX 5 A	5	20	TX 13 A
4	3	TX 4 A	4	3	TX 12 A
3	35	TX 3 A	3	35	TX 11 A
2	18	TX 2 A	2	18	TX 10 A
1	1	TX 1 A	1	1	TX 9 A
50	50	Discrete Output 8	50	50	Discrete Output 16
49	33	Discrete Output 6	49	33	Discrete Output 14
48	16	Discrete Output 4	48	16	Discrete Output 12
47	48	Discrete Output 2	47	48	Discrete Output 10
46	31	Discrete Input 8	46	31	Discrete Input 16
45	14	Discrete Input 6	45	14	Discrete Input 14
44	46	Discrete Input 4	44	46	Discrete Input 12
43	29	Discrete Input 2	43	29	Discrete Input 10
42	12	GND	42	12	GND
41	44	RX 8 B	41	44	RX 16 B
40	27	RX 7 B	40	27	RX 15 B
39	10	RX 6 B	39	10	RX 14 B
38	42	RX 5 B	38	42	RX 13 B
37	25	RX 4 B	37	25	RX 12 B
36	8	RX 3 B	36	8	RX 11 B
35	40	RX 2 B	35	40	RX 10 B
34	23	RX 1 B	34	23	RX 9 B
33	6	TX 8 B	33	6	TX 16 B
32	38	TX 7 B	32	38	TX 15 B
31	21	TX 6 B	31	21	TX 14 B
30	4	TX 5 B	30	4	TX 13 B
29	36	TX 4 B	29	36	TX 12 B
28	19	TX 3 B	28	19	TX 11 B
27	2	TX 2 B	27	2	TX 10 B
26	34	TX 1 B	26	34	TX 9 B

Anexo II

1. Funciones adicionales

1.1. Guardar

Esta función tiene como único fin la de permitir al usuario guardar los datos obtenidos tanto en el apartado de la gestión analógica como en la digital. Por otra parte, esta aplicación no podrá ser utilizada directamente por el usuario, es decir, que no podrá ver la interfaz.

Para poder satisfacer a los cuatro programas, es necesario poner el máximo número de variables de entrada y de salida. Para ello hay que fijarse en el programa Canal AI. El número máximo de entradas son treinta y dos. Dieciséis de ellos corresponden a la alimentación unipolar, mientras que los otros dieciséis a los diferenciales. La máxima salida que habrá será uno, que es por donde se extraerán los errores. Por tanto, se implementa en la función dos clusters de dieciséis arrays cada uno, y otro cluster de salida para el error. En cada array de entrada se introducirán los datos referidos a un canal específico.

Independientemente si los datos a guardar son del canal analógico de entrada o si son del canal digital o del contador, es necesario que todos los arrays de cada cluster de entrada se unan en un único array de dos dimensiones. Para ello, primero se separan todos los elementos de cada cluster en arrays independientes mediante la función “Unbundle”. Después se unen en un array de dos dimensiones a través la función “Build Array”.

Para guardar los datos, se utiliza a continuación, la función “Open_Create_Replace File” con la opción “create or replace” en la entrada “function (open:0)”. Después se coloca la aplicación “Write to Text File”, que permite escribir los datos. Antes de introducir los datos numéricos, es necesario colocar una cabecera que identifique los datos y la fecha que se guardó. Para crear la cabecera, se introduce en la función “Concatenate Strings” los datos citados anteriormente. Primero se coloca la fecha con la aplicación “Get Date/Time String”. A continuación, después de dos saltos de línea, se introduce la hora a la que se guarda mediante la aplicación anterior. Después se colocan tres saltos de línea.

El siguiente elemento a situar es la especificación de los datos. Como hay tres tipos de datos, tensión, bits y tiempo
, se coloca dicha especificación dentro de una estructura tipo “Case”, cada una en cada caso. Para escoger que tipo de dato es, se sitúa una variable donde se seleccionará la opción Canal analógico, si el programa es Canal AI, Canal digital, si por el contrario resulta ser el programa Canal Digital DI/DO, o Contador, si el programa es el Contador. Dicha variable se elegirá en cada programa específico.

A continuación se colocan otros tres saltos de línea para situar después un aviso que indique al usuario que no modifique ningún elemento. Finalmente,

después de tres saltos de línea más, se introduce otro texto que muestre que comienzan los datos.

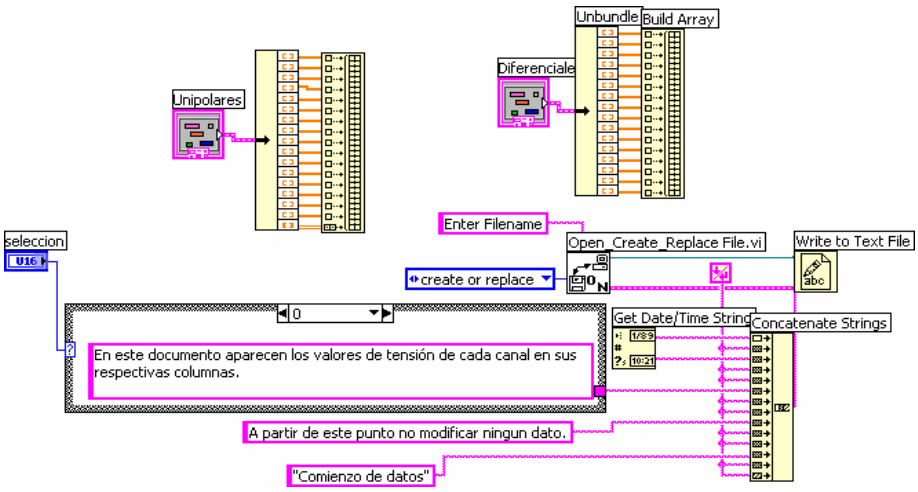

Fig. II.1 Esquema para introducir la cabecera y agrupar todos los elementos en un array.

A continuación se procede a explicar como se guardan los datos analógicos.

Los datos numéricos unipolares y diferenciales se introducen en la aplicación “Transpose 2D Array”. El resultado que se obtiene es el cambio de columnas por filas. De esta forma se obtienen los valores obtenidos en cada canal por filas. Los dos arrays, se introducen dentro de una estructura tipo secuencia (“Stacked Sequence Structure”). En la primera secuencia se colocan los canales unipolares usados, mientras que en la segunda se introduce la cabecera de los números de los canales diferenciales.

En la primera secuencia, se vuelve a poner la función “Concatenate Strings”. Después de un salto de línea, se introduce el nombre de los datos que se están escribiendo. Para este caso, “Alimentación Unipolar”. A continuación se procede a colocar el la palabra “canal” seguido del numero al que corresponde. Cada canal se separará mediante un espacio. La array unipolar se sitúa nuevamente dentro de otra estructura de secuencia (“Stacked Sequence Structure”). En este caso, el número de secuencia que hay son dieciséis. En cada secuencia se realiza una búsqueda para identificar en qué canales hay información y en cuales no. Por tanto, lo que se consigue con las dieciséis secuencias es recorrer las dieciséis posiciones posibles.

En la primera secuencia hay una estructura “Case” para determinar si la información que se transmite para guardar es digital o no a través de la variable “selección”. Si la variable resulta ser “Canal digital”, entonces se ejecuta la parte cierta. En caso contrario, la función realiza la operación del caso falso.

Dentro del caso falso, se implementa pone la función “Buscador” que permitirá determinar qué canal contiene datos. En el siguiente punto se

explicará exactamente como realiza la búsqueda. En este punto sólo se indicará los elementos de entrada y de salida.

En la función buscador, se introduce un número inicial, un carácter determinado distinto del espacio en blanco, otro número inicial que corresponderá al índice del array, y el array donde se buscarán los datos. A la salida se obtendrá un número de salida que volverá a ser introducido en la secuencia en el número de entrada, y el canal junto al número correspondiente en formato string. Este string se sitúa por una parte dentro de la aplicación "Concatenate Strings", y por otra, dentro de la "variable string IN". El valor de dicha variable se vuelve a colocar en la siguiente secuencia, en la entrada donde se a puesto antes un carácter inicial. Las siguientes secuencias, son exactamente iguales a la primera pero sin la estructura "Case".

La concatenación final se extrae fuera de la estructura de la secuencia para situarlo dentro de la aplicación "Write to Text File".

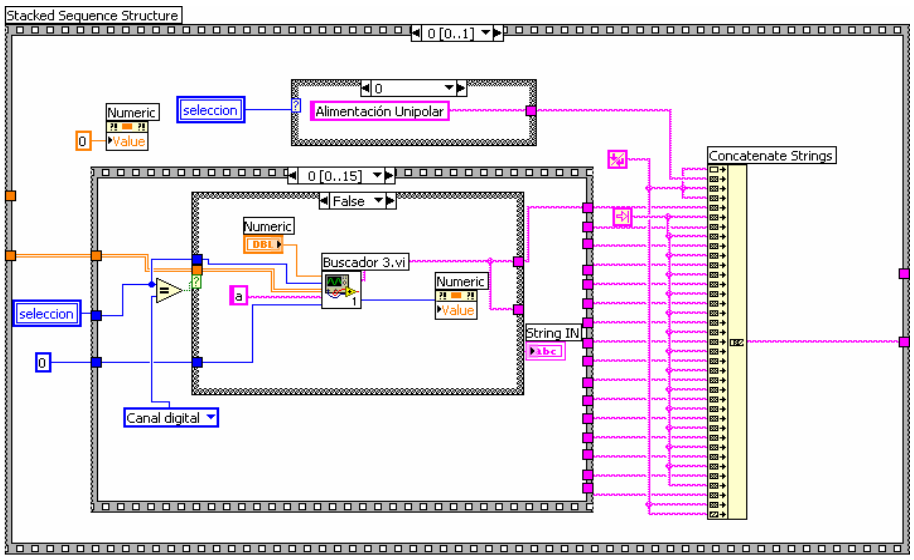

Fig. II.2 Estructura para introducir en la segunda cabecera el número del canal.

La siguiente secuencia es exactamente igual pero con dos diferenciales. La primera es que la otra estructura de secuencia ("Stacked Sequence Structure") se encuentra dentro de una estructura caso para determinar nuevamente si la información introducida corresponde a "Canal digital" o no. Si la condición es cierta, la estructura de secuencia ("Stacked Sequence Structure") no se ejecuta. Sin embargo, si es falso, entonces si. La segunda diferencia es que cuando se escoge la opción "Canal analógico", el string corresponde a "Alimentación diferencial". La cabecera obtenida de la función "Concatenate Strings", se sitúa dentro de la función "Write to Text File".

Una vez que se tiene la cabecera inicial definitiva, se procede a guardar los datos. Para ello, se elimina primero las columnas que no tengan información. A continuación, el array correspondiente a los datos unipolares, se introducen en una estructura de tipo "Case", mientras que el array con los números diferenciales, se sitúan en otra estructura de caso.

La estructura correspondiente a la alimentación unipolar, aparecen tres opciones, Canal analógico, Canal digital y Contador. Como en este caso se guardan datos analógicos, se describirá de momento el caso correspondiente a Canal analógico.

En él, el array unipolar se coloca dentro de un “Foor Loop “. Este permitirá pasar los datos del array por columnas. Dentro del “Foor Loop”, se introduce la columna del array dentro de la función “Index Array” y se escoge el primer elemento. Después, se compara con la constante –99999. Si no se cumple la igualdad, se ejecuta la parte falsa de la estructura “Case”, donde los datos son guardados en la variable “output array” a través de la aplicación “Insert Into Array”. En caso que sea cierta la condición, los datos de dicha columna no se guardan.

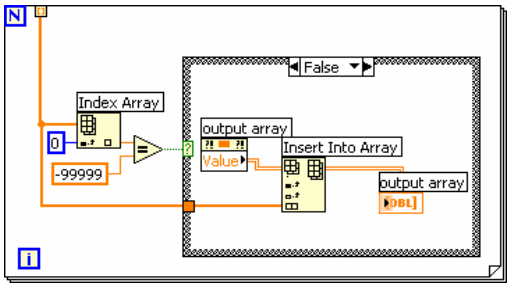

Fig. II.3 Esquema para discernir las columnas que tienen datos de las que no.

Para la estructura diferencial ocurre lo mismo, sólo que en este caso distingue si la opción escogida es Canal digital o no. En caso de que sea cierto, la estructura no realiza ninguna operación, mientras que si es falsa, se procede a repetir la misma operación que en el caso de los datos unipolares, con la diferencia que los datos se guardan en la variable “output array 2”.

El proceso de guardar los datos digitales es similar al de los datos unipolares. Sin embargo, existen unas cuantas diferencias. La primera se encuentra en la estructura de secuencia. En este caso, no pone ningún tipo de string adicional correspondiente al canal, simplemente deja un salto de línea más mientras que en el lugar donde se situaba “Alimentación Unipolar”, ahora se coloca “Canal Digital”. Para ello, en la primera secuencia, cuando la estructura se hace cierta, pasa un carácter de espacio en blanco que se coloca en la variable “String IN”. Esta a su vez, la introduce en la siguiente secuencia, dentro de la función “Buscador”. Al introducir el carácter de espacio en blanco, la función extrae en la salida “String Out” otro espacio en blanco. Además, la segunda secuencia, al ejecutarse la parte cierta de la estructura, no realiza ninguna operación.

A la hora de seleccionar los datos, la estructura que corresponde a los datos diferenciales, no realiza ninguna función al ejecutar la parte cierta. En cambio, en la estructura referente a los datos unipolares, los datos se introducen directamente en la función “Transpose 2D Array”, para cambiar de

posición los elementos. De esta forma lo bytes se consiguen ver en filas en vez de columnas. El problema de ponerlo en columnas es que si se guardan más bytes, es decir, columnas, que las que permite el fichero, éste automáticamente sitúa los bytes restantes debajo de los bits anteriores. Después de realizar la transpuesta, los valores se introducen dentro de la aplicación “Array Subset”. En ella, se selecciona las columnas comprendidas entre el índice cero y el dieciséis.

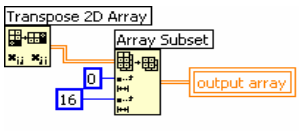

Fig. II.4 Estructura para obtener los datos referentes a las señales digitales.

En el caso de la opción “Contador”, las operaciones son las mismas que las del “Canal analógico”, sólo que en vez de obtener de la función la palabra “canal” junto al número correspondiente, se extrae el nombre de la medida en la cual pertenece la columna. Además, las palabras de “Alimentación, unipolar”, se intercambian por las de “Contador”, mientras que para “Alimentación diferencial” se sustituye por “Contador RC”.

El siguiente elemento que aparece, es una estructura de secuencia (“Flat Sequence Structure”). Tiene un total de tres secuencias.

En la primera secuencia, el primer elemento que aparece es una estructura “Case”. La ejecución de la parte cierta o falsa, depende de si ha ocurrido algún problema en las funciones anteriores. En caso de que sea cierto, los datos introducidos en la variable “output array”, no se guardan en un fichero. En el caso falso, dependiendo del tipo de datos, preguntará si se desean guardar los valores de la variable “output array”. En caso afirmativo, los valores son traspuestos para guardarlos en columnas. Después, se convierten todos los números en strings mediante “Array To Spreadsheet String”. Finalmente los datos son guardados en el fichero a través de la aplicación “Write to Text File”.

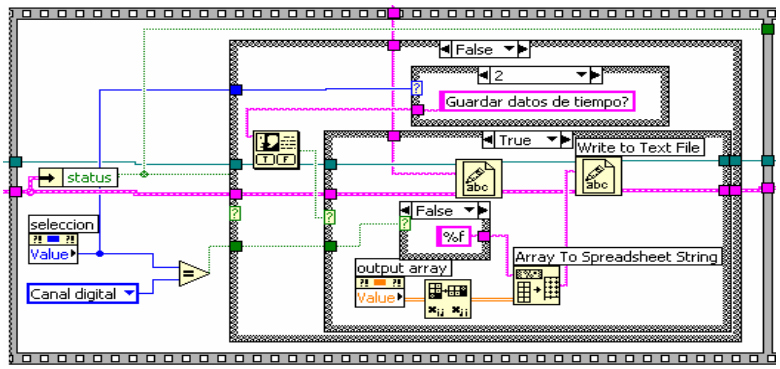

Fig. II.5 Esquema de la primera secuencia.

La segunda secuencia posee la misma estructura que la primera. De hecho, realiza la misma operación, pero en este caso, lo que guarda en el fichero son los valores de la variable “output array 2”.

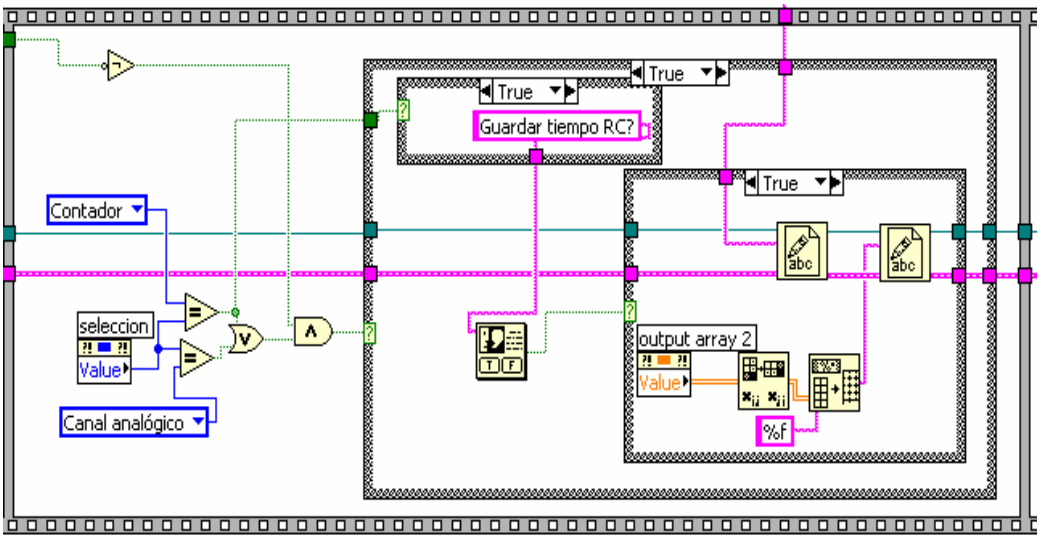

Fig. II.6 Esquema de la segunda secuencia.

La última secuencia permite reinicializar las variables “output array” y “output array 2”. Además, cierra el fichero abierto a través de la función “Close File”.

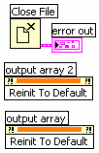

Fig. II.7 Esquema de la tercera secuencia.

1.2. Buscador

Como se ha dicho en el punto anterior, las variables de entrada son, “String IN”, es por donde se pasan los caracteres obtenidos en la secuencia anterior, “Numeric”, “Selección”, que permite mostrar si los datos corresponden al “Canal analógico” o no. En la variable “Array” se sitúan los valores que permitirán buscar los números adecuados de los canales o los nombres de medida del contador. La última variable corresponde al “index”.

Los valores de todas las variables se colocan dentro del bucle. Este se detendrá cuando el incremento de la variable “index” sea igual a dieciséis, que corresponde al número total de canales analógicos.

De los valores de la variable “Array”, sólo se coge el primer elemento de cada columna. Para ello, dicha variable se sitúa dentro de la aplicación “Index Array” y se introduce el valor de la variable “index” tanto a la entrada “col”, como “row”, cuyo valor inicial es cero. En cada iteración, se incrementa en uno el valor a colocar en la entrada “col”. A continuación verifica si el número obtenido es igual a –99999. En caso de que sea cierto, realiza la parte falsa en el que considera que la columna esta vacía y por tanto, extrae un carácter e blanco por la variable “String OUT”. Si no es así, ejecuta la parte cierta. En ella extrae por la variable anterior la concatenación de la palabra “canal”, junto con el valor de la columna acabada de incrementar, siempre y cuando, la variable “selección” corresponda a “Canal analógico”.

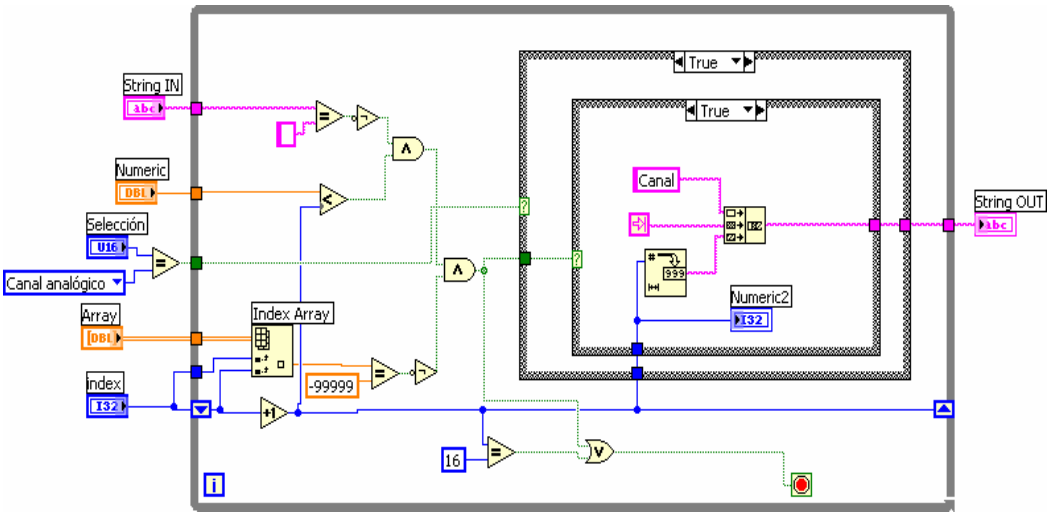

Fig. II.8 Esquema para determinar el nombre o numero de la columna.

En cambio, si la variable lleva la opción “Contador”, hace la función descrita en el primer párrafo de este punto. Sólo cabe destacar que si en la entrada de la variable “String IN” se coloca un espacio en blanco como carácter, considera que se han visto todas las filas y por tanto no realizará ninguna operación más de búsqueda.

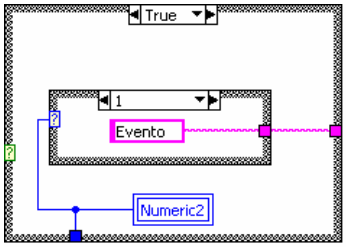

Fig. Anexo II.9 Estructura cierta.

1.3. Programa para guardar y cargar automáticamente direcciones y estructuras de palabras.

A diferencia de los programas anteriores, la estructura principal para éste es de tipo caso (“Case“). La razón es que este programa es para implementarse dentro de otro programa.

La tarea que desempeñará será, por una parte, cargar todas las estructuras de palabras que se hayan guardadas en un formato de texto, y por otra, guardar todas aquellas estructuras de palabras que se hayan definido antes de la puesta en marcha del programa.

El primer case que aparece, se compone por dos casos. El primero con el nombre de “create or replace” y el segundo denominado “open”.

En el primero, aparece un array de cluster, donde se introducen en cada cluster todos los elementos necesarios para configurar la estructura de una palabra. En la siguiente figura se visualiza su estructura.

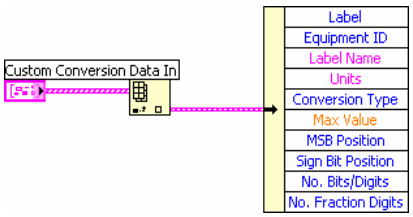

Fig. II.10 Variables dentro del cluster de conversión de datos.

Sobre este array es necesario saber que no pueden introducirse más de cien elementos. Esto es debido a que la librería vinculada a las funciones que permiten guardar las estructuras de palabras, “ar_def_custm_conv” y “ar_def_cstcvt_data”, no admite más de cien elementos.

El siguiente elemento que se aprecia es el “For Loop”. Permite procesar de uno en uno todos los elementos del array anterior que contengan algún elemento escrito, siempre y cuando se active la opción de desindexar los elementos del canal de entrada. Dentro de “For Loop” se separan los ocho elementos que se encuentran dentro de cada cluster a través de la función “Unblundle”. Todas las variables a excepción de “Label Name” y “Units”, se transforman en strings por medio de la aplicación “Number To Decimal String”.

Para que la transformación de ocho de los diez elementos en strings no ocupe una excesiva cantidad de espacio, es recomendable que se implemente con una estructura de secuencia (“Stacked Sequence Structure”) con ocho pasajes después de la función “Unblundle”. Dentro de cada pasaje, se introduce la función “Number To Decimal String”. De esta manera se convierten las ocho variables en un espacio reducido.

A continuación se concatenan entre paréntesis los nombres de las direcciones y las unidades a través de la función “Concatenate Strings”. Los datos convertidos anteriormente en strings se concatenan junto con el nombre de las unidades y de las direcciones. Cada elemento a concatenar se coloca en una fila diferente. Encima de estos elementos, se encuentra situado la palabra “datos” junto con el número de iteración que en ese momento realiza el “For Loop”. El string obtenido, se vuelve a concatenar nuevamente con el string nuevo que haya surgido del siguiente “cluster”. Cuando todos los elementos del array se procesen, el string final obtenido vuelve a concatenarse con las palabras “librería:” y “numero de entradas;”. Este último también se encontrará unido al número final de iteraciones del “For Loop”. Este número servirá después para saber el número de “clusters” guardados dentro del fichero.

Una vez obtenido la estructura deseada del documento final a grabar se procede a guardarlo. Para guardarlo, primero se introduce la función “Open_Create_Replace File”. En la entrada “function (open:0)”, se coloca la constante “create or replace”, que permitirá crear un archivo o guardar encima de otro. Para la entrada “prompt” se introduce la constante Enter “Filename”. La función que sigue es “Write To Text File”. Esta aplicación es la que permite guardar los datos mediante la inserción de estos en la entrada “text”. Finalmente, se cerrará el documento abierto usando la función “Close File”.

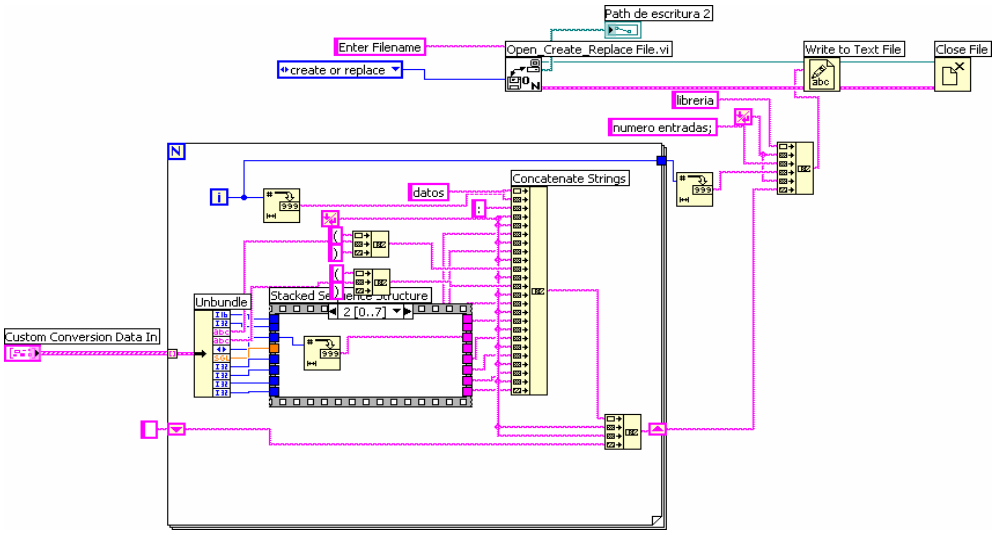

Fig. II.11 Esquema para guardar todos los elementos de cada estructura.

Una vez visto como se guarda las estructuras se continuará con la carga de estas. Al igual que en el caso anterior, se introduce la función “Open_Create_Replace File” con la diferencia que en la entrada Function (open:0) se une a la constante “open”. La siguiente aplicación es “Read From Text File”, que a partir de la cual se extraen todos los datos en formato string. Mediante la función “Close File”, se vuelve a cerrar el documento.

Cuando los datos son extraídos del fichero una parte se introduce dentro de un bucle, y otra parte se deposita entro de la aplicación “Match Pattern”.

Sirve para encontrar strings concretos a partir de una expresión introducida en la variable “regular expression”. En este caso se desea buscar el número que viene después de la palabra “número de entradas;”, ya que esta indica el número máximo de elementos que contiene la array anterior. Para poder obtener el valor, es necesario realizar dos búsquedas con la aplicación anterior. En la primera se coloca en la variable “regular expression” el carácter punto y como, y se extraen los datos por la salida “after substring”. Esto permite eliminar todos los datos que se encuentren delante del punto y coma, es decir, la palabra “librería” y “número de entradas”. En la segunda búsqueda, la expresión a introducir es “datos”, mientras que los datos se obtendrán por la salida “before substring”. El “string” obtenido permitirá indicar al bucle el número máximo de iteraciones que debe de hacer.

Dentro del bucle, los datos adquiridos de la función “Read from Text File” son introducidos dentro de la aplicación “Match Regular Expression”. Esta función es similar a “Match Pattern”, con la diferencia que esta permite buscar strings con expresiones más largas, es decir, mediante frases. La expresión a colocar dentro de la variable “regular expression”, es la concatenación de la palabra “datos”, junto con el número de iteraciones que se obtiene de cada ejecución del bucle y el carácter dos puntos. Como el número de iteraciones aumenta en cada ejecución, se pueden extraer a partir de esta expresión todas las estructuras guardadas anteriormente. Los datos adquiridos en cada iteración, se conducen por una parte a las funciones “Match Pattern”, y por otra parte a la función “Spreadsheet String To Array”.

En esta ocasión, las funciones “Match Pattern” se encargan de extraer los nombres de las direcciones y las unidades.

Para sustraer el nombre de la etiqueta sólo es necesario implementar dos veces la misma función. La primera función se encarga de buscar la apertura del paréntesis y de sacar en el indicador “after substring” los caracteres que vienen después de la apertura del paréntesis. La segunda función obtiene todos los datos que se encuentran delante del paréntesis de cerrado. Para ello se introduce el carácter anterior en la entrada “regular expression”, y obtener la información por la salida “before substring”.

El procedimiento a seguir para la extracción de las unidades es igual que antes, solamente que esta vez se ha de aplicar tres veces la misma función. La primera se usará para extraer datos que vengan después del paréntesis de cierre. Las dos restantes tendrán la misma utilidad que en el primer caso.

Los datos que se dirigen a la función “Spreadsheet String To Array”, son transformados a la salida de la función en un array de dos dimensiones escritos en una columna los valores numéricos. Puesto que sólo interesa la primera columna del array, se convertirá los datos en una array de una sola dimensión. Para ello se utilizará la función “Index Array”. En ella, se introduce una constante cero en la entrada “index (col)”. De esta manera se selecciona la columna para modificarlo en un array de una dimensión. A través de la entrada “array”, se coloca el array de dos dimensiones. En la salida subarray se obtienen los elementos modificados.

El array obtenido contiene en total cinco ceros. Los dos primeros se encuentran en las dos primeras posiciones del array mientras que los dos últimos se sitúan en el quinto y sexto lugar, y el quinto cero en la última posición. La razón de los ceros viene determinado por la aplicación “Spreadsheet String To Array”. Esta función no puede transformar en números decimales aquellos caracteres que no sean de tipo número, y por lo tanto lo muestra en forma de cero. Para visualizar mejor donde se encuentran los ceros, se muestra a continuación un ejemplo de una de las estructuras guardadas.

Ejemplo:

datos0:

165
2
(Custom Airspeed)
(Knots)
2
512
29
0
11
2

Se puede observar que en la primera fila, donde pone “datos0:”, se compone de caracteres que no contienen ningún número, a excepción del cero, con lo cual, la función citada anteriormente lo reconoce como un cero. Ocurre lo mismo en la segunda, quinta, sexta y última fila.

Para eliminar estos ceros, se coloca tres veces la función “Delete From Array”. La primera se sitúa en el primer pasaje de una estructura de secuencia (“Stacked Sequence Structure”). La segunda en el siguiente pasaje. La tercera en el pasaje final. La que se encuentra en el primer pasaje se introduce una constante de valor cero por la entrada “index”, y otra de valor dos por en la entrada length. De esta forma se elimina los dos primeros ceros. Además, todos los números se desplazan hacia las posiciones vacías que han dejado los ceros anteriores. En la segunda función la constante a colocar en la variable “index” es dos, mientras que en la entrada length, el valor es el mismo que en el caso anterior. Los ceros eliminados son los que se situaban en el ejemplo anterior en las filas cinco y seis. Para eliminar el último cero, se debe introducir una constante de valor ocho, en la aplicación del último pasaje.

Para finalizar, se debe seleccionar uno a uno todos los elementos que se encuentren dentro del array y depositarlo en la respectiva posición de la función “Bundle”. La posición que corresponde a cada elemento es la misma que

aparece en la figura de la página 41. En algunos casos es necesario cambiar la representación de los datos de DBL a "Integer".

La selección de cada uno de los elementos del array se realizará a partir de la función “Index Array”. Además, como son ocho elementos a separar, se ha optado por poner otra estructura de secuencia (“Stacked Sequence Structure”) con ocho pasajes. En cada pasaje se seleccionará un elemento en función del pasaje que esté ejecutando el programa En el primer pasaje se selecciona el primer elementos introduciendo el valor cero en la entrada “index”, mientras que en el último pasaje se coloca la constante siete para acceder al último elemento.

Para poder ser utilizado dentro de otra función, es necesario que disponga en su icono los siguientes elementos de salida. El array de cluster, donde se almacena la configuración de las estructuras, la variable que indica si se realiza la función de guardar o abrir y el error de entrada. La conexión se realiza de la siguiente forma. Primero se aprieta con el botón derecho en el icono situado en la parte superior de la derecha. A continuación aparecerá un menú. Seleccionar después la opción “Show Connector”. De esta forma aparecerán las terminales disponibles a conectar. Finalmente se aprieta dentro de uno de los terminales con la opción en el ratón de conexión, para después conectarlo a una de las variables anteriores. En el caso de elementos de salida son, el array de clusters donde se han cargado las estructuras de palabras y el error de salida. Para hacer visible la conexión se realiza la misma operación que en el caso anterior.

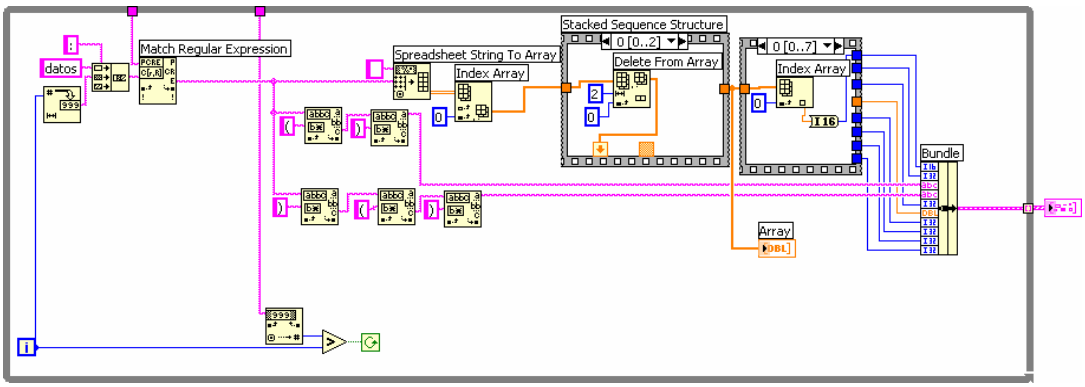

Fig. II.12 Estructura para cargar los datos de cada estructura guardada.

1.4. Conversión datos decimal a BCD

En este punto se pretende crear un módulo que simule la conversión de datos ingenieros a palabras binarias de estructura BCD.

Para convertir un número decimal a una palabra BCD, hay que tener en cuenta que hay que transformar en variable de tipo Unsigned de 32 bits cuatro elementos. Estos cuatro elementos son, la dirección (Label), la identificación fuente / destino (SDI) y la matriz de estado (SSM).

El valor del primer elemento se pondrá en una variable tipo Unsigned de ocho bits. A continuación, se transforma en un array de booleanas. Dicha operación se vuelve a repetir con el segundo elemento. El siguiente proceso a implementar es la concatenación de los dos datos anteriores de tal forma que los datos referidos a la dirección queden en primer lugar de los datos de SDI. Para ello se utiliza la función “Insert Into Array”, donde en la variable de entrada “array” se coloca la dirección transformada. En la entrada “subarray” se introduce el elemento correspondiente al SDI, mientras que en la entrada index, se pone el valor ocho.

En el caso de la transformación del valor numérico en un array de booleanas que se pueda concatenar con la array anterior, la estrategia consiste en separar un número entero en valores independiente. Para separar cada elemento, el número se divide entre diez. El resultado obtenido se redondea siempre hacia abajo mediante la función “Round To –Infinity”. A continuación se vuelve a multiplicar por diez. De esta forma se consigue que la primera cifra sea cero. Así, cuando se haga la diferencia entre el valor anterior y este. El valor obtenido es la primera cifra. Una vez obtenido la cifra, se cambia la representación de tipo DBL a Unsigned.

Antes de iniciar la separación en elementos independiente, es necesario saber la resolución de la que dispone. La resolución determinará que posiciones dentro del campo de datos corresponden a los valores representados y cuales se deben rellenar con ceros porque el sensor no es capaz de mostrar mayor resolución.

La representar dicha resolución se averiguará a través de las cifras decimales que tenga. Para ello, el valor numérico de la resolución se introduce dentro de un bucle. A continuación, se multiplica por una constante inicial de diez. El resultado obtenido se coloca dentro de la función “Expression Node”. Esta aplicación permite realizar operaciones que se encuentren escritas dentro de ella. Dentro de ella, se implementa la siguiente operación, X%1. Esta operación permite extraer el resto de de una división, en concreto, la de dividir la multiplicación entre uno. La variable de salida de dicha función es la resolución, pero con la coma desplazada un lugar a la derecha y sin el primer número decimal. Si a dicha variable de salida se compara con cero, se obtiene la condición de parada del bucle, es decir, el bucle continuara ejecutándose hasta que el cociente de la división sea cero.

Por otra parte, el valor de la constante diez se multiplica nuevamente por diez, y se transmite en la siguiente iteración mediante la función “Add Shift Register”. Cuando se vuelve a ejecutar la siguiente iteración, el resultado de la operación anterior, se multiplicará por el valor inicial de la resolución.

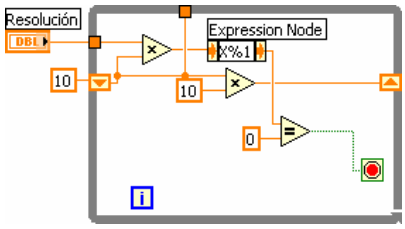

Fig. II.13 Esquema para obtener el orden de resolución

Cuando el bucle se detenga, el último valor obtenido de la multiplicación con la constante diez, se extraerá hacia fuera y se multiplicará por el valor absoluto del número a transformar.

El valor a transformar, primero es necesario eliminar el signo negativo, ya que este vendrá representado en el campo SSM. Para eliminarlo, se utiliza la función de valor absoluto “Absolute Value”. Una vez eliminado el signo, se multiplica por el resultado dado en el bucle anterior tal y como se explica en los párrafos anteriores.

El número resultante se coloca dentro de otro bucle. Por una parte se divide entre diez y se redondea hacia abajo utilizando la función “Round To -Infinity”. Por otra parte, se introduce dentro de la aplicación “Expression Node” con la expresión X%10.

El primer resultado permite eliminar el primer elemento del número. También indica si se han separado todos los números, comparando dicho resultado con cero. Si esta condición se cumple, el bucle se detendrá automáticamente.

El segundo resultado, se convierte a una variable de tipo Unsigned de ocho bits, y a continuación en un array de booleanas. El valor obtenido se vuelve a colocar dentro de otra función, “Insert Into Array”. Los datos se introducen en la variable de entrada “subarray” y se indican en la posición donde se quiera insertar a través de la entrada “index”. En la entrada “array”, se coloca el elemento donde se situarán los nuevos datos. Dicho elemento debe encontrase sin datos para no interferir en la conversión. A su vez, la array de salida se vuelve a introducir en la siguiente iteración mediante “Add Shifth Register”. La array anterior se coloca por la entrada de la función “Insert Into Array”. El nuevo elemento a introducir, debe situarse en la cuarta posición. De esta forma se eliminan los ceros innecesarios obtenidos al transformar el número anterior en variable Unsigned de ocho bits, y al mismo tiempo se colocan los cuatro primeros bits del número obtenido en esta iteración. Por tanto, el valor del índice que se sitia en la entrada “index” de la función “Insert Into Array”, se incrementará de cuatro en cuatro comenzando desde cero, por cada iteración ejecutada.

Una vez transformado se procede a concatenarlo con la dirección y con la SDI de la misma forma que se ha hecho anteriormente, pero con la diferencia que el valor de la constante para la entrada “index”, es diez.

El último elemento a concatenar es la SSM. El proceso es el mismo que en los casos anteriores. Nuevamente cambia la constante en la entrada “index”. Esta vez con valor veintinueve.

Finalmente se transforma la array resultante en una variable tipo Unsigned de 32 bits.

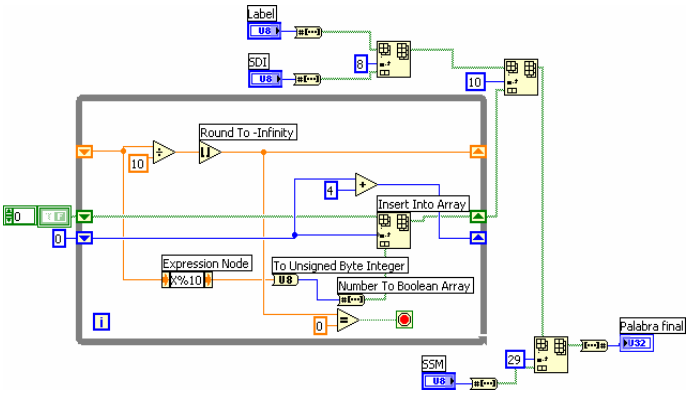

Fig. II.14 Estructura de la conversión ingeniero-BCD.

1.5. Conversión datos BCD a decimal

Al igual que en el caso 1.4, la intención es crear un programa que realice la conversión de palabras binarias de tipo BCD a datos decimales. Para ello, se intentará conseguir separar las palabras enteras en los cuatro elementos anteriores: dirección, SDI, dato numérico y SSM.

Primero se ha de introducir la palabra a convertir en una variable de tipo Unsigned de 32 bits, y a continuación convertirlo a un array de booleanas.

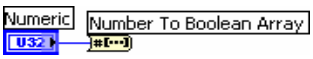

Fig. II.15Conversión de variable Unsigned 32 bits a array de booleanas.

El siguiente elemento que se pone, es una estructura de secuencia (“Stacked Sequence Structure”). En un principio no es necesario para que el programa funcione correctamente, sin embargo, las funcionalidades son más fáciles de localizar y reconocer.

En la primera secuencia, se extrae la dirección a través de la función “Array Subset”. Para escoger los ocho primeros bits se introduce en la entrada “index”, la constante cero, y en la entrada “length”, es decir la longitud del array de salida, la constante ocho, que representa las ocho posiciones de la dirección. Una vez que se haya obtenido la array deseada, se convierte a una variable Unsigned de ocho bits.

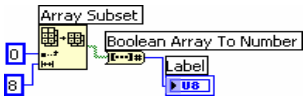

Fig. II.16 Conversión de array de booleanas a variable unsigned 8 bits.

En la segunda secuencia, se adquiere el elemento SDI. La operación es la misma que en el caso de la dirección. Pero con la diferencia que en la constante en la entrada “index” es ocho, mientras que en la de “length” es dos.

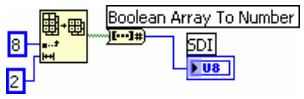

Fig. II.17 Conversión de array de booleanas a variable unsigned 8 bits.

En la tercera secuencia, se transforman el campo de datos a números decimales. Para ello, es necesario conocer los decimales de la constante de resolución. La misma razón que en el punto1.4. El proceso de extraer los decimales también es el mismo que en el apartado anterior.

La estrategia a seguir es similar a la conversión de datos ingenieros a datos BCD, con la diferencia de que lo que se extrae son variables booleanas de cuatro bits para ser representado en forma decimal.

La array resultante de la palabra de treinta y dos bits, se coloca dentro de un bucle. Dentro del bucle, igual que en las secuencias anteriores, se usa la misma función para extraer los bits de cuatro en cuatro. Para realizar esta operación, es necesario que el valor inicial de la entrada “index” sea diez, y se incremente por cuatro en cada iteración, mientras que en la entrada “length” se coloca una constante de valor cuatro. Los valores extraídos se convertirán primero en variable de tipo Unsigned y después en DBL.

Para concatenar los números de forma correcta, se comienza en la primera iteración a multiplicar el valor obtenido por uno. El resultado obtenido se transfiere a la siguiente iteración, mientras que la constante uno se multiplica por diez y se pasa también a la siguiente ejecución del bucle. En la segunda iteración se multiplica el siguiente resultado obtenido de la extracción y se multiplica por el resultado de multiplicar la constante uno por diez. De esta forma se consigue que el valor obtenido sean decenas. A continuación se suma el número convertido anteriormente con este resultado. De esa manera, se consigue concatenar dos números. El resto de números a concatenar seguiría el mimo proceso.

El bucle se detendrá cuando realice cinco ejecuciones. El porqué del número cinco viene dado por la estructura interna del campo de datos. Si el número total de posiciones que tiene el campo es veinte, y la subdivisión de

cada campo se compone de cuatro bits, a excepción del primero que son tres, sale un total de cinco subcampos a implementar.

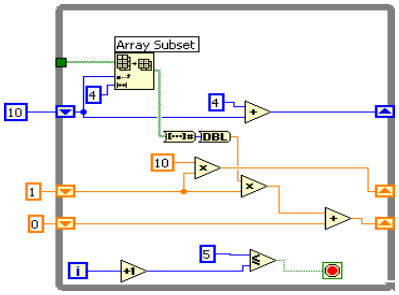

Fig. II.18 Esquema de conversión de BCD-Datos ingeniero

En la cuarta y última secuencia, se extrae la SSM con la condición que la constante de entrada “index” es veintinueve, y el valor en la entrada “length” ha de ser dos. Este último elemento, determinará el signo del número. Por tanto, se implementará con dos estructuras de caso (“Case”), La primera determinará si el valor obtenido es positivo o no. El segundo determinará si el número es negativo o no. Si se obtiene un valor positivo, se ejecuta la parte cierta, cuya función es pasar la constante uno fuera de la secuencia. Si no es positivo, pasa a ejecutar la parte falsa. Dentro de ella se encuentra el segundo bucle. Este realizará la parte cierta si el número es negativo. Su finalidad es extraer la constante menos uno fuera de la estructura. Si no es negativo, se ejecuta la falsa colocando fuera de la secuencia cero.

La constante extraída fuera de la estructura de secuencia, se multiplica por el resultado obtenido en la tercera secuencia.

1.6. Conversión datos decimales a BNR

Al igual que en el apartado 1.4 del Anexo II, para convertir los datos de decimal a BNR. Los elementos a tener en cuenta para la conversión son los mismos que en los dos puntos anteriores.

En este caso, lo único que cambia a la hora de convertir los datos es el campo de datos. El resto se realiza de la misma manera que en el apartado 1.50 del Anexo II. Por tanto, en este punto sólo se explicará la conversión del campo de datos de forma más detenida

Antes de comenzar hay que destacar que se debe tener en cuenta dos casos. El primero es cuando el número es positivo, y el segundo, cuando es negativo.

Para distinguir si el número es positivo o negativo, se pasa el valor del número por el comparador “Greater Or Equal To 0?” y se envía el resultado a la función “Select” Si la condición es cierta, dicha función transmitirá el número a transformar, sino, se le sumará la constante de la dirección específica. Por otra parte, para evitar que el número negativo puesto por el usuario se salga de la

escala de representación, se vuelve a comparar para saber si el resultado de la suma es mayor que cero. Si no se cumple la condición, se transmite cero.

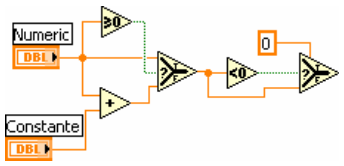

Fig. II.19 Esquema para determinar si es positivo o negativo.

Una vez obtenido el número a representar, se explicará a continuación la estrategia a seguir.

Se comienza dividiendo la constante entre dos. Si el resultado es más pequeño o igual que el valor a representar, entonces se representa bit uno en la primera posición del campo de datos de una palabra. Además, al valor numérico se le resta el cociente de la división anterior. El valor obtenido será el siguiente a comparar en la próxima ejecución. Además, el cociente resultante se vuelve a dividir entre dos durante la ejecución del bucle. El nuevo valor se pasa también en la iteración en la siguiente iteración para volver a realizar la comparación.

Sin embargo, si el resultado de la división es mayor que el número, entonces se pone cero. El resultado obtenido de la división, se vuelve a dividir entre dos. El nuevo resultado es comparado nuevamente por el valor numérico inicial.

Esta idea representada en el Labview consiste en colocar un bucle por donde se introducen los datos numéricos a transformar y la constante de la dirección dividida entre dos. El bucle continuará ejecutándose hasta que recorra las dieciocho posiciones del campo de datos.

Dentro del bucle, aparecen dos estructuras de caso (“Case”). Una externa y otra interna. La interna se ejecutará cuando la externa sea falsa. En un principio, se transmite el valor falso para la estructura externa. Dicha estructura será cierta cuando el valor de la resolución escrita sea más grande que la división de la constante entre dos. Cuando la condición se cumpla, la estructura cierta se ejecutará en la siguiente iteración. En ella, se transmitirá únicamente ceros para transformarlos después en booleanas.

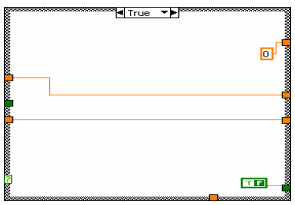

Fig. II.20 Esquema de la estructura cierta externa.

La estructura interna será cierta cuando se cumpla que el número a convertir sea mayor que la división de la constante entre dos. Cuando se ejecute, transmitirá una constante de valor uno. Además, la constante se dividirá nuevamente entre dos, mientras que el dato numérico se le sustraerá el valor de la división obtenida en la anterior iteración y se volverá a introducir en la siguiente ejecución del bucle.

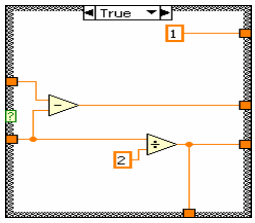

Fig. II.21 Esquema de la estructura cierta interna.

Cuando se cumpla la condición falsa, se enviará hacia fuera la constante cero. Al igual que en el caso anterior, se vuelve a dividir entre dos la constante.

El valor final extraído de las dos estructuras, se convierte en una variable de tipo Unsigned de ocho bits, para después volverlo a convertir en un array de booleanas. Dicha array, se sitúa dentro de la función “Index Array”, donde se introducirá el valor cero en la entrada “index”. De esta forma se obtiene sólo el primer elemento de la array. A continuación, se indexa a la salida del bucle para poder conseguir de esta forma concatenar todos los valores de cada posición del campo de datos y obtenerlos en forma de array de booleanas.

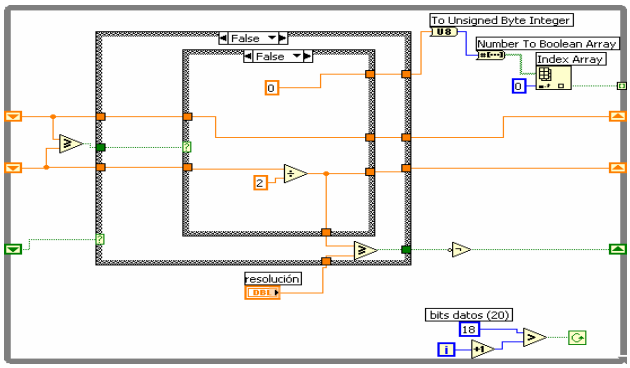

Fig. II.22 Estructura de la conversión ingeniero-BNR.

Para finalizar con el programa, solamente faltaría concatenar el array de direcciones junto con el de la SDI, el del número, y el de la SSM. Como se ha comentado al principio de este apartado, este proceso, se describe en el apartado 1.4Cabe destacar una diferencia, y es que en este caso, la posición del bit 29 se concatena a parte de los datos numéricos, ya que indica el signo del número. Por tanto, el bit veintinueve se concatenará con la función “Insert Into Array” con el valor veintiocho en la entrada “index”.

1.7. Conversión datos BNR a decimal

Para completar el conjunto de módulos de conversión, se implementará en este apartado un programa que transforme las palabras binarias de tipo BNR a datos ingenieros.

El programa es idéntico que el que se explica en el punto 1.5 del Anexo II. La única variación que se aprecia, es la secuencia donde se transforman los datos.

Antes de transformar el campo de datos, es necesario saber que signo lleva dicho campo. Para ello, se sitúa la palabra entera en la función “Index Array”, escogiendo únicamente la posición 29. Como en los puntos, se realiza la misma operación de selección, colocando en este caso, la constante veintiocho. La variable booleana obtenida permitirá a la función “Select”, enviar una constante cero, si es falso el resultado, o menos uno, si es verdadero. El resultado se multiplica por la constante de la dirección y se introduce dentro de un bucle. Cuando se multiplica por cero, significa que el valor a transformar es positivo, mientras que si se encuentra multiplicado por menos uno, es negativo.

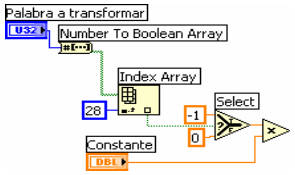

Fig. II.23 Esquema para determinar si la estructura BNR es positiva o no.

La estrategia de implementación es similar al apartado anterior, pero en este caso algunas operaciones son inversas.

Como en el caso anterior, el bucle se parará cuando recorra las dieciocho posiciones del campo de datos. Por otra parte, a la hora de adquirir los valores de cada posición a través de la aplicación “Index Array”, se comienza desde la ultima posición, es decir, la posición 28. Esto requiere que en la entrada “index” se introduzca el valor inicial veintisiete, y que en cada iteración, dicho valor se disminuya en uno. La razón de esta implementación viene dado por el hecho de que cada posición esta multiplicado por la constante y dividido entre 2^n , correspondiendo valor uno de la n para la última posición, mientras que para la primera el valor dieciocho. Se puede observar que es más fácil introducir una constante inicial de valor dos dentro del bucle e ir multiplicando por dos en cada iteración, que no comenzar con la constante 2^{18}.

Una vez extraído el valor correspondiente de cada posición, se convierte a través de la función “Boolean To (0,1)” en una variable de tipo Unsigned en el que adquirirá valor uno, o cero. Por otra parte, la constante de la dirección se divide entre dos. El resultado obtenido, se multiplica por el valor transmitido por la función “Boolean To (0,1)”. El número resultante, se transfiere en la siguiente

iteración para sumarse al nuevo valor obtenido de la operación descrita anteriormente. A su vez, el cociente de la división entre la constante entre dos, se pasa a la siguiente ejecución del bucle para volverse a dividir de nuevo.

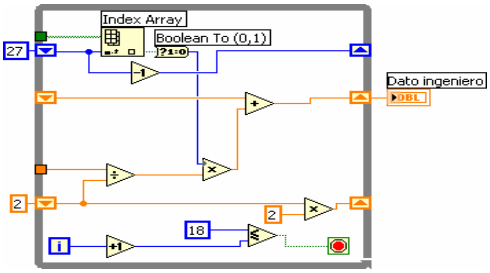

Fig. II.24 Esquema de conversión BNR-ingeniero.